A STUDY OF THE EFFECT OF THE OBJECT-ORIENTED PROGRAMMING PARADIGM ON PROGRAM DESIGN COMPLEXITY USING ADA, MODULA-2, AND C++

By

BUDY TJAHJO

Bachelor of Science in Arts and Sciences

Oklahoma State University

Stillwater, Oklahoma

1985

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1989

A STUDY OF THE EFFECT OF THE OBJECT-ORIENTED PROGRAMMING PARADIGM ON PROGRAM DESIGN COMPLEXITY USING ADA, MODULA-2, AND C++

Thesis Approved:

Thesis Adviser

Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to thank many people who gave me encouragement and advice during the preparation of this thesis. I thank Dr. K.M. George for his encouragement, guidance, and patience from start to finish. I also thank Dr. M. Samadzadeh who taught me so much about experimental design. Without his advice and guidance, this thesis would not have been possible. I appreciate Dr. G.E. Hedrick's serving on my graduate committee.

To the student volunteers who participated in the experiment, I express gratitute. Without their involvement, this study would not have been possible. I thank Teo Ming Fah and Becky Bishop for their help and advice throughout the thesis preparation. Finally, I thank my parents, Hendro Tjahjo and Ida Sutjiapti Tjipto for their encouragement and support.

TABLE OF CONTENTS

1.SAS is a registered trademark of the SAS Institute, Inc., Cary, NC 27512, USA.

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

D	Difficulty metric (Halstead)
DA	Cox's data abstraction metric
DH	Cox's data hiding metric
E	Effort (Halstead)
H&K	Henry and Kafura's data sharing metric
L	Program level (Halstead)
n_1	Number of unique operators
n_2	Number of unique operands
n	Vocabulary metric ($n = n_1 + n_2$)
N_1	Total number of operators
N_2	Total number of operands
N	Vocabulary metric ($N = N_1 + N_2$)
Ne	Estimated N metric
TSO	Cox's time sequence of operations metric
v(G)	Cyclomatic number
V	Volume metric

CHAPTER I

INTRODUCTION

Statement of the Problem

The object-oriented programming technique has proved its worth as a tool for helping develop large and complicated programs, but few people agree on what it means. Although Smalltalk-80 and Ada differ greatly, some programmers call both of them object-oriented languages [16]. Most computer scientists agree that in order to use the object-oriented programming technique, one should choose a language that provides necessary features to support the technique [7].

The purpose of this study is to evaluate the impact of the object-oriented programming technique on novice programmers' program design complexity by using Ada and C++. To evaluate the novice programmers' performance, the experts' program design complexity is compared to the novices' program design complexity. The program design complexity is measured by using Halstead's software science [14], McCabe's cyclomatic complexity [14], Henry and Kafura's data sharing complexity [14], and Cox's surface area metrics [16].

The experimental data consisted of eight object-oriented designs chosen from several books. Three groups of programmers -- Ada experts, Modula-2 experts, and C++ experts -- designed the programs. These expert programmers wrote the books from which the eight designs were selected. The novice programmers were Oklahoma State University seniors and graduate students. Because of time and resource constraints in the controlled environment, the novices wrote only four of the selected programs in Ada and C++. A percentage method and an extrapolation method were used to measure partial programs.

The study first overviews the object-oriented programming technique and explains its usefulness in Ada, Modula-2, and C++. The study then discusses Halstead's software science, McCabe's cyclomatic complexity, Henry and Kafura's data sharing metric and Cox's surface area metric. Chapter IV describes the data collection and the experimental design for the study in more detail.

Chapter V evaluates the programs written by the expert and novice programmers and the metrics produced after applying the measurements. This chapter mainly discusses the linear relationship between the metrics. Since each program produced fifteen different metrics, many combinations of correlation coefficients could have been evaluated. In the expert data analysis section of this chapter, the correlation coefficients of the experts' Ada programs are analyzed followed by the experts' C++ programs and then the

experts' Modula-2 programs. The section also evaluates the correlation coefficients between two programs in the different languages. Besides the correlation coefficients, the mean values of the metrics are also compared.

Novice data analysis section of chapter V evaluates the metrics produced by applying the measurements to the novices' programs. This section describes the percentage and extrapolation methods used to measure the partial programs. Similar to the expert data comparison, the novice data comparison basically involves evaluation of the correlation coefficient of the novices' Ada metrics, novices' C++ metrics. The mean values of the novices' metrics are also evaluated.

Expert/novice data analysis section of chapter V discusses the comparison of experts' metrics to novices' metrics. Since the number of novice and expert programmers differed, as well as the number of problems they implemented, an extraction and duplication technique was applied to balance out the metrics. This technique is discussed in this section. Unlike the previous sections, this section compares the corresponding metrics of the same language used by the expert and novice programmers. For example, the program length metric of the experts' Ada program is compared to the program length metric of the novices' Ada program.

Literature Review

Cox [16] discusses object-oriented programming in general and briefly compares the object-oriented features of Ada and C++. In this study, surface area metrics are the only metrics that directly measure the complexity of an object-oriented design. The metrics are derived from Cox's definiton of surface area which Gannon, Katz and Basili [23] also support. A general study of object-oriented programming and language can be found in Cargill [11], Cox [16], Cunningham and Beck [18], Halbert and O'Brien [26], Mitchell, Urban, and McDonald [32], Rascoe [37], and Snyder [42].

Booch [7] gives pertinent Ada language references. He discusses examples reflecting good Ada style. This book is valuable for this study because of its discussion of object-oriented techniques used in Ada. Other studies involving the object-oriented technique for Ada are Booch [3 and 4], Buzzard and Mudge [10], Katwijk [28], Krogdahl [29], and Unger [47].

Strostrup [43] provides a comprehensive reference for C++. Since C++ is an object-oriented language, the example programs use the object-oriented technique. Wiener [54] gives a more comprehensive discussion of the involvement of C++ in object-oriented techniques. Dewhurst and Stark [19], Trickey [45], Unger [47], and Wiener and Pinston [54] also contain studies of the object-oriented technique in C++.

Most researchers refer primarily to Wirth [55] for Modula-2 information. Even though Wirth does not discuss the object oriented technique, he designs many examples in a style similar to object-oriented programming. Oktaba and Berber [34], Pomberger [35], Wegmann [49], and Wiener and Sincovoc [53] also researched the Modula-2 involvement in the object-oriented technique.

The software measurement literature was reviewed to find criteria to compare the expert programs to the novice programs. Conte, Dunsmore, and Shen [14] reviewed many software metrics including Halstead's software science, McCabe's cyclomatic complexity, and Henry and Kafura's data sharing complexity which were used in this study. Ramamurthy and Melton [36], Reynolds [38], Crawford, McIntosh, and Pregibon [17], Li and Cheung [30] and Reynolds [39] write about other studies involving software science and the cyclomatic number.

One advantage of the object-oriented technique is to cut down surface area, which is the number of things that must be understood and properly used for one programmer's code to function correctly in combination with another programmer's code [16]. Thus, it is necessary to use the surface area measurement to compare the programs. Cox [16] writes that the surface area increases with the increases of information hiding, abstract data types, and time sequence of operations. Gannon, Katz and Basili [23] also mention the

importance of the surface area measurement.

Reynolds [38] developed metrics to measure the complexity of partial programs. His later study about the partial metrics system was also reviewed [39]. Wiedenbeck [50] reports expert/novice differences in programming skills. Her experimental design is similar to the experimental design in this thesis.

Basic Definitions

Abstract data type: a model that encompasses a type and associated set of operations [54].

Class: the data structures and methods that implement objects as a private type. See abstract data type.

Constructor: the method that defines how a new object is constructed when it appears in initialization statements.

Cyclomatic complexity: the software metric designed to measure the number of "linearly independent" paths through a program.

Destructor: the method that destroys or deallocates an object allocated by a constructor.

Encapsulation: a technique that lets the user of an object see only the methods that are available from that object, but not how those methods are implemented.

Information hiding: data element names, data type names, or function names of an object that are not visible for the user of that object.

Inheritance: a tool for organizing, building, and using reusable objects.

Module: a Modula-2 program control structure that constitutes a fence around the data, data types, and procedures declared in it. It keeps these hidden from the outside world.

Object-oriented programming: a method that lets one work with problem-domain concepts rather than operator/operand concepts.

Opaque data type: a Modula-2 data type that allows its representation to remain hidden to the user of the definition module and is first fixed in the implementation module.

Operator/operand: a concept of how hardware works.

Overloading: an identifier that has several alternative meanings at a given point in the program text.

Packages: Ada's fundamental program units that support the software principles of data abstraction and information hiding.

Private type: a type whose structure and set of values are clearly defined but not directly available to the user of the type.

Reusability: a concept of object-oriented programming to use an an existing object for creating a new object.

Software complexity: a characteristic of the software interfaces which influences the resources another system will expend or commit while interacting with

the software.

Software crisis: difficulty in managing software development.

Software metrics: tools to characterize the essential features of software quantitatively, so that classification, comparison, and mathematical analysis can be applied.

Software science: Halstead's family of metrics that are functions of the counts of operators or operands.

Surface area: the number of things that must be understood and properly dealt with for one programmer's code to function correctly in combination with another's.

CHAPTER II

OBJECT-ORIENTED PROGRAMMING

Theoretical Background

Developing efficient, reliable, maintainable, and understandable software systems is a very complex task. Such a system frequently consists of tens of thousands of lines of code and is built by more than one person. The software crisis is the reflection of the problem of developing this massive software system [7]. Booch [7] discusses the nature and cause of the crisis in detail. Object-oriented programming is a technique that tries to control the crisis.

The decomposition of a system using object-oriented programming is based upon the concept of an object. The system mainly contains objects and messages rather than data and functions as in the conventional method. An object has a set of operations that perform on the object. The messages are the interaction between the objects or a request for an object to perform one of its operations.

Unger's [47] example of the object integer is a good representation for understanding the concept of an object. An integer may be represented by a data structure which

consists of a sequence of zeroes and ones which are interpreted as in one's complement arithmetic. The operations on the object integer are =, +, *, and /. Thus, the abstract type integer has a data structure which is used to represent the object integer and it has a set of operations defined for the object integer [47].

Using object-oriented programming requires an unconventional program development technique. The key is to design objects so that each clearly represents a single concept. This means that the programmer must focus on the following questions [7]:

- How are these objects created?
- Can these objects be copied and/or destroyed?
- What operations can be done on such objects?

If such questions have no good answers, the concept probably should not be an object in the first place, and perhaps more thought should be given to the problem.

Object-oriented programming centers around several major concepts, namely data abstraction, encapsulation, and information hiding. The major defect of the conventional method of programming is the scope and visibility that the key data structures have with respect to the surrounding software system. The implementation of many important functions depends on the key data structures. If any changes are made in one or more of these key data structures, the fall-out effects on the software system cannot be avoided [54]. Object-oriented programming concepts help produce

software that is more tolerant to change.

Encapsulation and information hiding bind data and procedures tightly together and limit the scope and visibility of the functions that can manipulate the data. They minimize interdependencies among separately written modules by defining strict external interfaces [5]. An abstract data type is a set of objects and operations whose implementation is hidden so that the user sees only the objects and operations as they manifest themselves through the application of the operations to the object [32]. The binding of the data with an associated set of functions that can manipulate the data is called encapsulation. The inaccessibility of the internal structure of the underlying data is called information hiding [54].

Even though inheritance is not a necessary feature of object-oriented programming and almost nonexistent in this thesis, it is briefly discussed for its desirable properties. Inheritance is a tool for organizing, building, and using reusable objects. Without inheritance, every object would be a free-standing unit, each developed from the ground up [16]. It allows programmers to reuse all or parts of an existing object in constructing a new object. In Smalltalk, an object-oriented language [54], inheritance forms the basis of an entire programming environment in which a hierarchy of objects is available to the user. Most Smalltalk programs are constructed by sending messages to objects from existing objects or creating objects derived

from existing objects and sending messages to these objects [54]. Snyder [42] discusses the type and the importance of inheritance in detail. In the following sections, Ada, Modula-2, and C++ features are discussed for their use of the object-oriented programming technique.

Object-Oriented Programming Using Ada

Although Ada may not fit all definitions of an object oriented language, it does have the object-oriented programming concepts mentioned above. To implement data abstraction, Ada provides packages and private types. Packages in Ada come in two parts, the specification and the body. The specification part specifies objects and operations whose implementations are hidden in the body part of a package. Thus, by seeing the specification part, one object can interact with another without knowing the body part. When designing a system, a user's only concern is with the specification part. The benefits of the packages not only enforce the data abstractions but also reduce the scope of the change upon the system because the replacement of the implementations in the body part will not affect the other part of the system.

Booch [5] applies the following features in the object-oriented programming technique:

- Classes of objects are denoted by packages that export private or limited private types.
- Objects are denoted by instances of private or

limited private types or as packages that serve as an abstract state machine.

- Object state resides either with a declared object (for instances of private or limited private types) or in the body of a package (in the case of an abstract state machine).
- Operations are implemented as subprograms exported from a package specification. Generic formal subprogram parameters serve to specify the operations required by an object.
- Variables serve as names of objects. Aliases are permitted for an object.
- Visibility is statistically defined through unit context clauses.
- Task and task type may be used to denote actor objects and classes of objects.
- Derived types can be used to denote a form of inheritance.

Consider, for example, the Ada package TREES, whose specification is shown in Figure 1. The user of the object TREE need only to consider this specification which is used as a user guide. The abstract data type in the example is TREE. The word "private" means that an object of type TREE cannot be manipulated except as a parameter to pass to one of the routines defined in the specification. This information hiding concept helps to eliminate the programming error outside the object TREES.

```
generic
     type ITEM is private;

package TREES is
     type TREE is private;
     type CHILD is (LEFT, RIGHT);
     NULL_TREE : constant TREE;

     procedure CLEAR         (THE_TREE      : in out TREE);
     procedure CONSTRUCT     (THE_ITEM      : in ITEM;
                              AND_THE_TREE  : in out TREE;
                              ON_THE_CHILD  : in CHILD);
     procedure SWAP_CHILD    (THE_CHILD     : in CHILD;
                              OF_THE_TREE   : in out TREE;
                              AND_THE_TREE  : in out TREE);
     function IS_NULL        (THE_TREE      : in TREE)
                               return BOOLEAN;
     function ITEM_OF        (THE_TREE      : in TREE)
                               return ITEM;
     fumction CHILD_OF       (THE_TREE      : in TREE;
                              THE_CHILD     : in CHILD)
                               return TREE;
     OVERFLOW      : exception;
     TREE_IS_NULL  : exception;

private
     type NODE;
     type TREE is access NODE;
     NULL_TREE : constant TREE is null;
end TREES;
```

(Adapted from Software Engineering with Ada, by Booch, G., Menlo Park, CA, Benjamin/Cummings, 1983.)

Figure 1. Specification for the package TREES.

```
with TREES;     -- make the specification visible

package body TABLES is
     package WORD_TREE is new TREES(WORDS.WORD);
     THE_TABLE: WORD_TREE.TREE;

     procedure START is
     begin
          WORD_TREE.CLEAR(THE_TABLE);
     end START;

     procedure INSERT (THE_WORD: in WORDS.WORD;
                IN_THE_TREE: in out WORD_TREE.TREE) is
         TEMPORARY_WORD : WORDS.WORD;
         TEMPORARY_TREE : WORD_TREE.TREE;
     begin
         if WORD_TREE.IS_NULL(IN_THE_TREE) then
                 TEMPORARY_WORD := THE_WORD;
                 WORD_TREE.CONSTRUCT(TEMPORARY_WORD,
                                     IN_THE_TREE,
                                     WORD_TREE.LEFT);
         else
                     .....
                     .....
         end if;
     end INSERT;

     <other operations for object TABLES>


end TABLES;
```

(Adapted from Software Engineering with Ada, by Booch, G., Menlo Park, CA, Benjamin/Cummings, 1983.)

Figure 2. The implementation of package TABLES which uses object TREES.

Object TREES is a generic package which means that it allows the user to pass parameters to the object TREES. These parameters can be data objects, types, or subprograms [10]. Figure 1 shows that the object TREES lets the user define the type ITEM which is the type of an object that the user can swap between object TREES or insert inside an object TREES. The generic concept makes the data abstraction and information hiding more complete which results in a much more reusable object [10].

Figure 2 shows how the object TREES is used in the application program. Object WORD_TREE is declared as object TREES with the WORDS.WORD as the type for object ITEM . Even though a user can use the type TREE and type CHILD, he cannot directly manipulate the data object with these types. For example, the user cannot manipulate the data object TABLE (Figure 2) except as a parameter in one of the object TREE operations.

A form of inheritance in Ada can be created by using derived types. A new object can be derived from the existing object that exports a nonprivate type and then builds on top of the existing object. The new object inherits all the operations from the parent object. The new objects can have new operations, replace operations, and hide operations from the parent object [5]. The trade-off for this flexibility is giving up the advantages of information hiding and encapsulation since the bind between the data object and the operations is no longer necessary.

Object-Oriented Programming Using Modula-2

As in Ada, Modula-2 was not designed to be an object-oriented language but has the necessary features to use the object-oriented programming technique in Modula-2 programming. In fact, by adopting the object-oriented programming technique, a Modula-2 programmer can dramatically increase the modularity of his/her application [49]. Modula-2 provides a module to encapsulate code and data and hence gives the programmer data abstraction. As a package in Ada, a module in Modula-2 splits into two parts--a definition part and an implementation part. The definition part consists of the data objects and the operations that manipulate the data objects. The implementation part provides the implementation of the operations defined in the definition part. The user needs only to view the definition part which is also the user guide to use the object defined in the module.

Wegmann [49] gives the conventions below to adopt the object-oriented programming technique in Modula-2:

- A class (object) is a module with a definition and an implementation part. The definition part declares all the types needed for the data object and the operations on the data object. The implementation part specifies the types and the operations.
- An instance (data object) is a dynamically

allocated data area - a RECORD. The implementation part provides procedures to create or destroy the data object.

- The methods (operations on the data objects) are PROCEDURES.

Figure 3 is the definition part of object STACKS which is used to describe how the object-oriented programming technique can be applied in Modula-2. Object STACKS exports type stack and the operations newStack, push, pop, and IsEmpty so that the user may declare object of type stack and use the operations to manipulate the object. Since type stack is an opague type (the internal structure is hidden), the user will not be able to use other operations to manipulate objects with type stack. This technique satisfies the definition of an abstract data type. The internal structure of type stack is defined in the implementation module. As in Ada, by having separate parts (definition and implementation), Modula-2 makes it possible to have strong binding between data objects and the operations used to manipulate the objects. This satisfies the encapsulation concept. By hiding the internal structure of type stack, Modula-2 makes it impossible for the user to manipulate objects with type stack except with the provided operations. This satisfies the information hiding concept.

Modula-2 can also have a form of inheritance (subclassing). Following the conventions stated above, the

inheritance is implemented by initializing a new object with the existing object [49]. The new object can override the operations it wants to change. Figures 5 and 6 illustrate how inheritance is used in Modula-2. Object NewComplex inherits all the methods implemented in the Object Complex (Figure 4) except that it overrides the operation GetPut [49]. This means operation NewDispose of object NewComplex operates exactly the same as operation NewDispose of object Complex. As in Ada, this flexibility has disadvantages. The user has to be able to access the data object of type Complex to be manipulated by new operations in Object NewComplex. This somewhat destroys the binding between the data object and the operations of object Complex.

```
DEFINITION MODULE STACKS;

    EXPORT QUALIFIED Stack, NewStack, Push, Pop, IsEmpty
    TYPE Stack; (* opaque type *)

    PROCEDURE NewStack(VAR s: Stack);

    PROCEDURE Push(s: Stack; elem: INTEGER;
                   VAR done: BOOLEAN;

    PROCEDURE Pop(s: Stack; elem: INTEGER;
                   VAR done: BOOLEAN;

    PROCEDURE IsEmpty(VAR s: Stack): BOOLEAN;

END STACKS.
```

(Adapted from Software Engineering and Modula-2, by Pomberger, G., Princeton, NJ, Petrocelli Books, 1985.)

Figure 3. The definition part of Module STACKS.

```
DEFINITION MODULE Complex;

    IMPORT Number;
    EXPORT QUALIFIED NewDispose, GetPut,
                     ObjectId  , Object, classId;

    PROCEDURE NewDispose(VAR objectId : Number.objectId;
                             new      : BOOLEAN        ;
                         VAR re,im    : REAL           );


    PROCEDURE GetPut    (VAR objectId : Number.objectId;
                             get      : BOOLEAN        ;
                         VAR re,im    : REAL           );

    TYPE
        ObjectId = POINTER TO Object;
        Object   = RECORD
                       c       : Number.ClassId;
                       re,im   : REAL;
                   END;
    VAR
        ClassId : Number.ClassId;

END Complex.
```

(Adapted from Software Engineering and Modula-2, by Pomberger, G., Princeton, NJ, Petrocelli Books, 1985.)

Figure 4. The definition part of Module Complex.

Object-Oriented Programming Using C++

Wiener et al.'s [54] definition of C++ "C with class" indicates the importance of the role of class in C++. A class is an abstract "user-defined" data type [43]. When used in object-oriented programming, class consists of private data and public operations. Consider Figure 7, to

create an object, a variable, Node, is declared as TreeNode. Only the operations of Node can manipulate the private data. The implementation of the operations is hidden from the user. The binding between the data and the operations of object Node represent the encapsulation concept. The inaccessibility of the private data represents the information hiding concept.

```
DEFINITION MODULE NewComplex;

    IMPORT Number;
    EXPORT QUALIFIED NewDispose, GetPut,
                     ObjectId  , Object, classId;

    PROCEDURE NewDispose(VAR objectId : Number.objectId;
                             new      : BOOLEAN        ;
                         VAR re,im    : REAL           );


    PROCEDURE GetPut    (VAR objectId : Number.objectId;
                             get      : BOOLEAN        ;
                         VAR re,im    : REAL           );

    TYPE
        ObjectId = POINTER TO Object;
        Object   = RECORD
                       c       : Number.ClassId;
                       re,im   : REAL;
                   END;
    VAR
        ClassId : Number.ClassId;

END NewComplex.
```

(Adapted from Software Engineering and Modula-2, by Pomberger, G., Princeton, NJ, Petrocelli Books, 1985.)

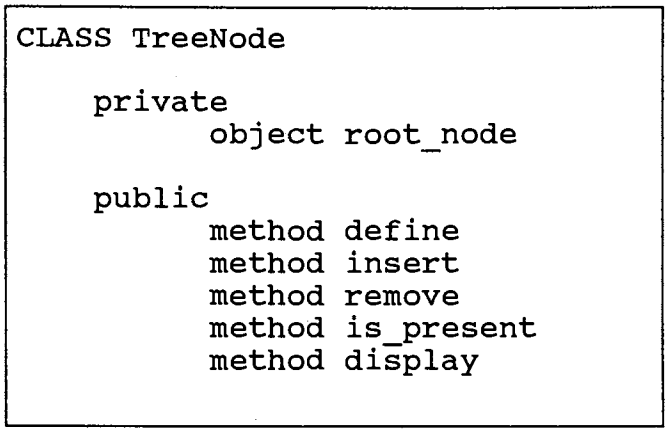
Figure 5. The definition part of Module NewComplex.

```
IMPLEMENTATION MODULE NewComplex;

    IMPORT Complex;
    (* other data objects declared here   *)

    PROCEDURE GetPut     (VAR objectId : Number.objectId;
                              get      : BOOLEAN        ;
                           VAR re,im   : REAL           );
    BEGIN

           < new implementation of GetPut >

    END GetPut;

END NewComplex.
```

(Adapted from Software Engineering and Modula-2, by Pomberger, G., Princeton, NJ, Petrocelli Books, 1985.)

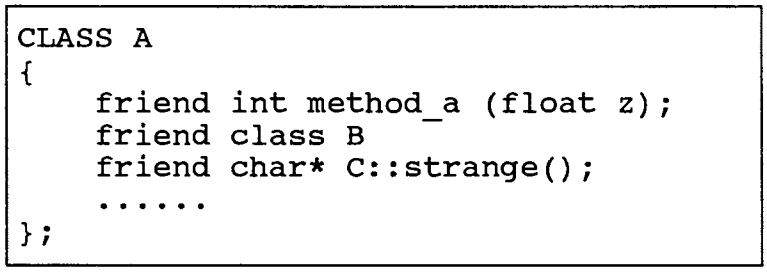
Figure 6. The implementation part of Module NewComplex.

C++ inheritance is possible by constructing a class derived from an existing class (parent class). Inheritance means allowing the derived class to access the private data of its parent [54]. This will destroy the advantage offered by information hiding. But, unlike Ada or Modula-2, C++ provides a more secure type of inheritance. A derived class in C++ has to have an authorization to access the private data of its parent.



(Adapted from An Introduction to Object Oriented Programming and C++ by Wiener, N., New York, NY, Addison-Wesley, 1988.)

Figure 7. The depiction of class tree.



(Adapted from An Introduction to Object Oriented Programming and C++ by Wiener, N., New York, NY, Addison-Wesley, 1988.)

Figure 8. The depiction of friend construct.

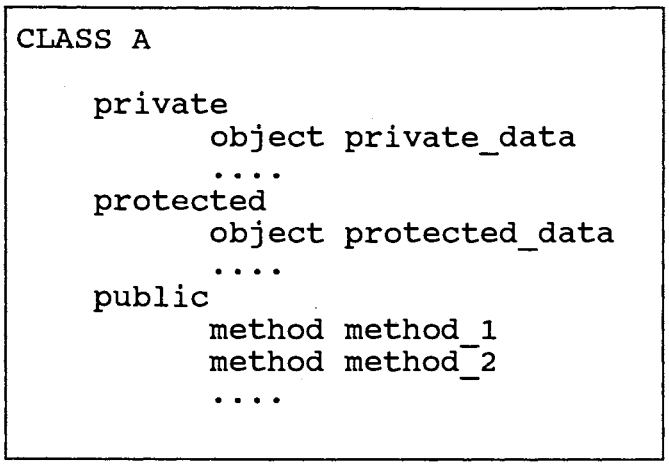


Figure 9. The depiction of protected data.

(Adapted from An Introduction to Object-Oriented Programming and C++ by Wiener, N., New York, NY, Addison-Wesley, 1988.)

There are two ways to obtain the authorization required to access the private data [54]:

(a) by declaring the derived class or a method in the derived class as a friend of the parent class. Figure 8 illustrates that the operation method_a, all of the operations of the derived class B, and operation strange of the derived class C can access the private data of class A;

(b) by declaring the data inside the parent class as protected data. Figure 9 illustrates that all of the derived class of the parent class A can access the protected data.

Discussion

Data abstraction, encapsulation, and information hiding are the key concepts of object-oriented programming. This chapter discussed the possibilities of applying object-oriented programming in Ada, Modula-2, and C++ program development. The package features of Ada, the module feature of Modula-2, and the class feature of C++ provide the data abstraction. The separation between the definition and the implementation part of a package, module, or class provide encapsulation and information hiding.

Even though inheritance is not a necessary feature of object-oriented programming, it is possible to have a form of inheritance in Ada, Modula-2, and C++. Inheritance makes an object much more reusable. The trade-off of this advantage is destroying the concept of information hiding. C++ provides a secure way to use inheritance by declaring the derived class as a friend of the parent class or by declaring protected data.

Many more features in Ada, Modula-2, and C++, such as genericity or overloading, make these languages even more useful in maintaining complex software. However, it is not the objective of this study to explore all of the features available in these languages. The main purpose is to demonstrate how these languages help novice programmers create an object-oriented program by comparing their program design complexity to the experts' program design complexity.

CHAPTER III

SOFTWARE METRICS

Theoretical Background

"The goal of software engineering is to produce software that is efficient, reliable, adaptable, maintainable, and easily usable" [14]. Object-oriented programming is a technique that tries to meet this challenge. As mentioned in the previous chapter, the main purpose of this study is to investigate how object-oriented programming can help novice programmers write efficient programs. This is why software metrics are needed in this study. They measure the novice programs' complexity objectively. Conte, Dunsmore, and Shen [14] state that "an objective, or algorithmic, measurement is one that can be computed precisely according to an algorithm. Its value does not change due to changes in time, place, or observer." Expert programs are also measured to be compared to the novices'.

Software metrics are mostly applied to the development process or to the software product. For this reason, most of the metrics are classified as process metrics or product metrics [14]. As defined in Conte, Dunsmore, and Shen [14],

"process metrics quantify attributes of the development process and of the development environment" and "product metrics are measures of the software product". This study uses both process metrics and product metrics for data analysis and comparison.

Conte, Dunsmore, and Shen [14] discuss three techniques to gather data. They are software analyzers (executable programs that measure the complexity of an analyzed program), report forms (information forms to be completed by analysts and programmers to help analyze the data), and interviews (ways to help the programmers fill in the report forms in order to understand the information required). In this study, all of these techniques are applied.

Researchers conduct exploratory study and confirmatory study. In exploratory study, the outcome of an experiment is unknown. In confirmatory study, the researchers try to confirm the outcome obtained in an existing study [14]. Since no study like this is known to exist and there is no hypothesis set, by definition this study is an exploratory study.

A large number of software metrics were investigated for inclusion in this study. Many of them are applicable to measure the complexity of the investigated programs. The software metrics below were chosen because they are easily measurable and expected to be useful for analysis and comparison.

Halstead's Software Science Metrics

All software science measures are functions of the four basic metrics n_1 (number of unique operators), n_2 (number of unique operands), N_1 (total occurrences of operators), N_2 (total occurrences of operands) (sm). Halstead proposed the following metrics to measure various aspects of software:

(a) Program Volume, $V = N * \ln(n)$. Halstead suggested that the human mind use a binary search to choose operators or operands in writing a program. The outcome is the number of decisions made by a programmer to finish a program. $N = N_1 + N_2$ and $n = n_1 + n_2$.

(b) Potential Volume, $V^* = (2+n_2^*) \ln(2+n_2^*)$. Halstead suggested that a program is composed of a group of procedures and operands as the input and output parameters. In the potential volume formula, 2 represents the procedure name and symbols that separate the procedure name from its parameters (sm1). n_2^* represents the number of input or output parameters. An operand used as input and output parameters is counted as two operands.

(c) Program Level, $L = V^* / V$. "Any program with volume V is considered to be implemented at the program level L" [14]. The value of the program level ranges from 0 to 1. The higher this value the less complex is the program.

(d) Difficulty, $D = 1 / L$. Halstead suggested that the difficulty of a program is the inverse of the program level.

(e) Effort, E = V / L = V * D. Rationally, it takes more effort to write a more difficult program or a bigger program. Therefore, Halstead formulates effort as dependent on the program volume and the program difficulty.

Even though some researchers dislike software science's "theoretical underpinnings (or the lack thereof)", many researchers agree that software science is the most comprehensive theory yet attempted of the software development process and continue to find these metrics useful as a basis for size and effort models [14].

McCabe's Cyclomatic Complexity Metric

McCabe suggests that the complexity of a program depends on the number of "linearly independent" paths throughout a program flow graph [14]. A linear program, a program without if-then-else constructs or loops, is considered the least complex. The metric value for the least complex program is one. McCabe formulates the cyclomatic complexity metric as V(G) = e - n + 2, where e is the number of edges and n is the number of nodes in a flow graph. This formula is derived from the concept that for each if-then-else construct or loop decision add one to the complexity number. This formula can be simplified as V(G) = DE + 1 where DE is the number of decisions. Figure 10 shows how a flow graph is measured. Since there is a relationship between this metric and the testability and maintainability of a program, this metric is much more useful for this

study.

Henry and Kafura's Data Sharing Complexity Metric

The concept behind this metric is that the complexity of a program increases with the number of data shared between procedures. Data shared between procedures can be the parameters of a procedure or global variables. Henry and Kafura [14] propose the following variant of metrics to measure the complexity of data sharing in a procedure:

(a) fan-in * fan-out

(b) $(\text{fan-in} * \text{fan-out})^2$

(c) $S_S * (\text{fan-in} * \text{fan-out})^2$

where fan-in is the number of procedures that send data to the procedure being measured either directly or indirectly (global variables). Fan-out is the number of procedures that receive data from the procedure being measured either directly or indirectly. S_S is the number of lines in the procedure. This metric is important in this study because it explores the importance of the information hiding concept in object-oriented programming.

Cox's Surface Area Metric

Cox [16] defines surface area as "the number of things that must be understood and dealt with for one programmer's code to function correctly in combination with another's". Surface area increases with the number of names that are

visible at the interface in a package, a module, or a class. This includes variable names, data type names, and function or procedure names. Surface area also increases with each type of dependency between these names. For example, a requirement that a function take two arguments, the first of type complex and the second of type float, increases the number of things that the consumer must get right to use this function successfully [16]. Requiring the consumer to know that an object must be allocated, then initialized, then accessed increases the number of things a consumer must get right, and thus increases the surface area [16].

From the above definition there are three metrics that can be derived from the surface area. They are:

(a) Information hiding metric, IH = NV + NP + NT where NV is the number of variables, NP is the number of procedures, and NT is the number of data types visible for the user. The assumption is the more visible an object, the more complex or difficult the object to be used [16];

(b) Data abstraction metric, DA = number of unique data types required for a user to understand to use the methods inside the object to be measured. The assumption is the more data types to be understood to use the object, the more difficult the object is to be used [16];

(c) Time sequence of operations, TSO = number of steps (operations) taken to start using the object. The assumption is the more steps needed, the more complex the object is. All of these metrics measure the visible part of an object.

There are the definition part of a package of Ada, a module of Modula-2, and a class of C++.

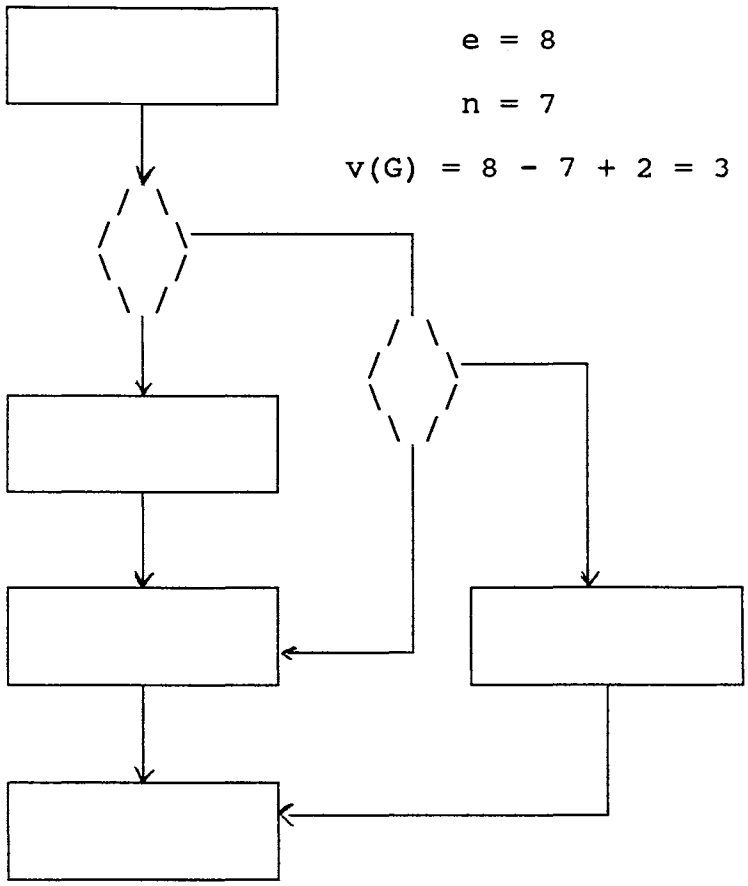


Figure 10. An example of a flow graph.

Experimental Design

It is possible to evaluate the program development process by conducting a controlled experiment. The following questions must be answered before starting an experiment [14]:

(a) Are the subjects novices or experts?

(b) Are the subjects prepared for the experiment?

(c) What treatment will prepare them?

Many biasing factors can occur during the experiment. Conte, Dunsmore, and Shen [14] list a number of factors that should be minimized. They are:

(a) External events such as suggestion, recommendation, encouragement, or discouragement from the outside which may affect the subjects' performance;

(b) Internal changes such as improving or deteriorating ability which may affect the subjects' performance;

(c) Changes of equipment required to do the experiment which may affect the subjects' performance; and

(d) In dividing the subjects into groups, a random method should be applied.

Experimental design deals with the best way to choose a design format so that the results of the experiment can be compared and analyzed. The following are the three experimental designs discussed in Conte et al. [14]:

(a) The Pretest-posttest design. It evaluates the differences between the results before and after the treatment. The results measure the subjects' performance. The treatment improves the subjects' performance.

(b) The Posttest-only design. As the name implies, it is similar to the pretest-posttest design without conducting a pretest experiment. So the results mainly measure the

subjects' performance after the treatment.

(c) Counter-balance design. This design technique gives all of the subjects all the treatments available in the experiment. So in this type of experiment, the number of results produced by each subject is equal to the number of treatments that the subject was exposed to.

Basili, Selby, and Hutchens' paper [3] is used as a guide to define and classify this study. They discuss four phases of an experimental process as follows:

- The study definition phase containing six parts: 1) motivation, 2) object, 3) purpose, 4) perspective, 5) domain, and 6) scope.
- The second phase of the experimental process is the study planning phase. The aspects of the experiment planning phase are: 1) design, 2) criteria, and 3) measurement.
- The third phase of the experimental process is the study operation phase. The operation of the experiment consists of 1) preparation, 2) execution, and 3) analysis.
- The fourth phase of the experimental process is the study interpretation phase. The interpretation of the experiment consists of 1) interpretation context, 2) extrapolation, and 3) impact.

Discussion

Conte, Dunsmore, and Shen [14] provide many references for metrics that can be used to measure program complexity. The problem is to choose the best metrics to obtain numbers that can be compared and analyzed for this study. Halstead's software metrics and McCabe's cyclomatic complexity metric were chosen because they are used in most studies concerning software complexity. Basili suggests that there is a relationship between these metrics and that the maintenance process makes these metrics an attractive choice [38]. Henry and Kafura's data sharing metric is selected because the issue of data sharing is an important one in object-oriented programming. The factors that influence the surface area of an object are translated into the information hiding metric, data abstraction metric, and sequence of operations metric. The formulas for these metrics are produced by Cox's definition of surface area.

Experimental design was reviewed in this chapter. How a controlled experiment should be conducted was discussed. The biasing factors that may occur during the experiment are listed. These factors need to be minimized, if elimination is not possible, to get a valid result. Three types of experimental designs reviewed were pretest-posttest, posttest only, and counter-balance designs. The next chapter discusses how the counter-balance design is applied in the controlled experiment part of this study. The classification

of the experiment according to the framework defined by Basili, Selby, and Hutchens [3] is also discussed in the next chapter.

CHAPTER IV

DESIGN OF THE EXPERIMENT

About the Experiment

A counter-balance or latin square design was chosen for this experiment because of the relatively few available subjects. The experiment can be represented by

	t_1	t_2
Novice group 1	X_1O	X_2O
Novice group 2	X_2O	X_1O

The notation used is taken from Conte, Dunsmore, and Shen [14]. The symbol X represents the exposure of a group to a certain treatment. The symbol O represents the measurement after the treatment. The symbol t represents the treatment.

In the experiment, eight programs were selected from several Ada and C++ text books [2, 7, 21, 33, 35, 43, 46, 48, 52, 55, 56]. From the eight programs, four were translated into problem specifications (Appendix A). Eleven computer science students implemented the problem specifications in Ada and C++ in a controlled environment. Along with the problems, the students were also given the main drivers

(Appendix B, Appendix C) and were asked to implement the problem so that the program satisfied the main driver's requirement. The main driver made sure that the students would write the programs in the object-oriented programming style.

The students were randomly divided into two groups. On the first day of the experiment, one group implemented the problems in Ada and the other group implemented the problems in C++. On the second day, the two groups implemented the same problems in the other language.

Approximately three hours of object-oriented programming style in Ada and three hours of C++ lecturing were given to the students as treatments. Table 1 shows the schedule of the treatments. Even though most of them had been writing in Ada and C, no student had been exposed to the object-oriented programming style of Ada and C++ before.

A total of forty-four programs produced by the students were measured by the metrics discussed in the previous chapter. At the end of the experiment, a posttest questionnaire (Appendix D) was given for discussion and analysis purposes.

The expert group is not included in the counter-balance design above because the programs produced by this group were collected from the textbooks written by the experts. This group was included in the study so that their programs could be measured and compared to the novices' programs. There were eight programs written in the object-oriented

programming style for the three languages measured, i.e., Ada, C++, and Modula-2. The twenty-four expert programs which, as stated above, were extracted from various textbooks, were implemented and debugged, so that executing implementations were available.

Experiment Framework

The motivation of this study is to understand the effect of the object-oriented programming technique on developing programs. As mentioned in the beginning, the purpose of this study is to evaluate the novice programmers' program design complexity after applying object-oriented programming. The product of the first part of this study is the object-oriented programming programs created by the novice and expert programmers.

The domain of this study is discussed under the sections subjects and tasks below. According to the definition given by Basili, Selby, and Hutchens [3], this study is a multiproject variation study since the scope is examining objects across a single team and a set of projects. The rest of this chapter discusses the experiment planning. The experiment operation and interpretation is discussed in the next chapter.

Subjects

Eleven computer science senior and graduate students from Oklahoma State University participated in this experi-

ment (three graduates and eight seniors). They were randomly divided into two groups: novice group 1 and novice group 2. They were called novices since no one had been exposed to the object-oriented programming style. Novice group 1 consisted of five students and novice group 2 consisted of six students. On the first day of the experiment, each group was asked to implement task 1 which consisted of three problems. One group wrote the task in Ada and the other group in C++. On the second day of the experiment, each group was asked to implement task 2 which consisted of one problem in the other language.

Tasks

In selecting the programs for the tasks, several Ada, C++, and Modula-2 programs were collected from textbooks written on the subject of object-oriented programming style. From among those programs, the programs that performed similar functions in the three languages were selected. The following are the selected programs:

(a) Spelling checker

(b) Search tree table

(c) Matrix

(d) Complex numbers

(e) Rational numbers

(f) Link list

(g) Stack

(i) Queue.

Because of the time constraint, four programs were selected for task 1 and task 2. Task 1 consisted of programs (c), (d), (f), and (i). Task 2 consisted of program (b). Since a Modula-2 compiler was not available, the programs were implemented in Ada and C++. The more descriptive problem statements for each task can be found in Appendix A.

Task Schedule

Some of the tasks above were implemented before the experiment began. From the experience, the subjects were given three hours to finish each task. So, a total of six hours, excluding the treatments, were required to complete the experiment. The following time tables show how the experiment and the treatments were scheduled.

TABLE I

TREATMENT TIME TABLE

Treatment	Subject	Time	Before the exp.
Ada in general	Groups 1 and 2	3 hours	2 week
C++ in general	Groups 1 and 2	3 hours	1 week
OOP in Ada	Group 1	30 min.	a half hour
OOP in C++	Group 2	30 min.	a half hour
OOP in Ada	Group 2	30 min.	a half hour
OOP in C++	Group 1	30 min.	a half hour

TABLE II

EXPERIMENT TIME TABLE

Subject	Day 1	Day 2
Group 1	Task 1 in Ada	Task 2 in C++
Group 2	Task 1 in C++	Task 2 in Ada

Treatments

Before the experiment days, the subjects were prepared to enable them to write a program in Ada and C++. Since they were all senior and graduate students, they were expected to learn the structure and the syntax of Ada and C++ quickly. The subjects were also taught how to compile and link Ada and C++ programs. At this point the subjects were not supposed to be familiar with the object-oriented programming style. So the package feature of Ada and the class feature of C++ were not presented to them. Approximately two hours of teaching (in the form of a classroom lecture) were given to the subjects for each language. To minimize the external factors influencing the subjects, the experiment was conducted as soon as the time allowed (more precisely, four days after the last lecture).

On the first day of the experiment, novice group 1 was taught the importance of the class feature of C++ and its

properties--such as data hiding, encapsulation, and data abstraction, and how they should be applied in the experiment to produce object-oriented programming style code. The same is true for the novice group 2 except for the fact that they were taught the package feature of Ada. For approximately thirty minutes, the object-oriented programming style was taught to each group. Then, the subjects were given the problem statements (task 1) and asked to write them in the object-oriented programming style in the language they had just been taught.

On the second day of the experiment, the same routine was conducted as the first experiment except that novice group 1 was exposed to the package feature of Ada and novice group 2 was exposed to the class feature of C++. After the thirty-minute treatment, they were given task 2 and asked to write the task in the object-oriented programming style using the language they had just been taught.

The interval between the first and the second experiment was two days. During this time, they had no idea what language they would be asked to use. The short interval between the two phases of the experiment was an attempt to minimize the internal factors influencing the subject factors, such as their increased knowledege of the object-oriented programming style for the language they were going to use. The treatments were designed in this way to standardize the subjects' knowledge of the object-oriented programming style.

Equipment and Technical Assistance

All of the subjects developed their Ada programs on Dec Vax 11/780 system and their C++ programs on the Perkin Elmer 3230 system (now a concurrent XF610 system). They were all familiar with the editors available in those systems. Ada programmer's guides and C++ manuals were provided for the subjects. A technical assistant was in the laboratory during the experiment to enforce the validity of the experiment as well as to help the subjects understand what they were expected to do in the experiment. The assistance provided by the technical assistant was limited to answering the questions regarding the problem statements and compiling the programs.

Observations

To measure the experts' and the subjects' programs, three analyzer programs were developed. Those three programs read Ada, C++, and Modula-2 programs and produced Halstead's and McCabe's metrics. Henry and Kafura's and Cox's surface area metrics were measured manually. The next chapter is devoted to the analysis and comparison of the calculated metrics. In addition, the subjects' behavior was monitored during the experiment (consulting the Ada and C++ manuals, encountering problems, dealing with the problems etc.).

Discussion

In this chapter, a counter-balance experiment was defined and discussed. The experiment was conducted with the subjects divided into two groups and given treatments. The subjects, the task, and the treatments were discussed in detail. The experiment was designed only for novice programmers.

CHAPTER V

ANALYSIS AND EVALUATION

Eleven complexity metrics are produced by feeding the programs into complexity analyzers and four metrics are produced manually. Those eleven metrics are Halstead's and McCabe's metrics while the last four metrics are Henry and Kafura and Cox's surface area metrics. The three analyzers evaluate C++, Ada, and Modula-2 programs. The metrics are then fed into SAS, Statistical Analysis System, to produce scatter plots and correlation coefficients.

Basically, the analyzers count the operands and the operators of their corresponding programs. Keywords, operators, and reserve words are considered operators. The rest are considered operands. From the number of operators and operands, Halstead's metrics and McCabe's metric are generated. Halstead's metrics are the functions of operators and operands so the metrics can be directly applied. McCabe's metric is produced by adding all the occurrences of the basic condition keywords (WHILE, IF, ELSE IF, UNTIL, etc) plus one. A condition statement such as "if (a == b) || (c !=d)" in C++ is considered to have two basic condition keywords.

A method derived from Henry and Kafura's definition of

data sharing complexity is used to measure Henry and Kafura's data sharing metrics. The measurement procedure is outlined below.

A) Calculate the fan-in of the program based on the fan-in of its constituent modules. The fan-in of a module is the sum of all of the external variables and the number of modules that pass data to the module under consideration [14]. The module in this case is a procedure or a function. The fan-in of a program is the average of the fan-in of the modules existing in the program. The average is calculated by the sum of the fan-in of all modules divided by the number of modules. The average value is chosen to express the fan-in of a program since the fan-in value represents the module not the program.

B) Calculate the fan-out of the program (see fan-in). The fan-out of a module is the sum of all of the external variables and the number of modules to which data is passed [14]. As the fan-in, the fan-out of a program is the average of the fan-out of the modules existing in the program.

C) The Henry and Kafura metric is (fan-in X fan-out).

The measurement of Cox's surface area metrics involves only the visible part of an object which is the specification part. Information hiding, data abstraction,

and time sequence of operations are measured in the following manner:

A) The information hiding metric of an object is the sum of all variables, procedures, functions, and data types inside the specification part of the object. All the information inside the private area is also considered hidden so the information is not measured even though it is inside the specification area.

B) The data abstraction metric of an object is the number of data types required to understand how to use the object. This includes the user data types and the types of the parameters of the operations of the object.

C) The time sequence of operation metric of an object is the number of operations taken to use the object. Creating, accessing, and destroying are examples of sequence of operation. For example, in most objects created in Ada language, it is necessary for a user of the object to allocate memory for the object, then initialize or access data inside the object, and eventually destroy the object.

Measurement Scales

There are four different scales of measurement discussed in Conte, Dunsmore, and Shen [14]. The nominal

scale classifies the data. As stated by Conte, et al., "The only property of importance is equality and inequality" [14]. For example, the object-oriented programming style or top-down design can be important information in the analysis of complexity metric. So the data that obtained from object-oriented programs or from top-down programs are from nominal scales. The ordinal scale ranks the data. For example, levels of programmers' experience, levels of program complexity, or levels of programming language are from ordinal scales. The interval scale expresses the difference between two data in a measurement unit such as inch, meter, pounds, etc. [30]. The ratio scale uses the ratio of two data items. For example, number of language's keywords are from a ratio scale because it could be said that Ada language's keywords are twice as much as C++ language's keywords. The ordinal scale is chosen because it is more practical for this experiment.

The correlation coefficient between two variables is used to analyze and compare different metrics. It is widely used to evaluate the strength of a relationship between two sets of measures [14]. There are two kinds of correlation coefficients, parametric and nonparametric. The parametric correlation coefficient assumes that the relationship between two variables is linear [14]. The nonparametric correlation coefficient assumes only that the two variables use ordinal scales. In this study, the parametric Pearson correlation coefficient is used because it is believed that

there is a linear relationship among the selected metrics. They are produced by SAS by feeding the C++, Ada, and Modula-2 complexity metrics into it. The value of the coefficient is between zero and one. The values near zero imply a lack of linear relationship, the values near one imply a strong linear relationship, and the negative values imply a negative linear relationship [14].

The formula for the Pearson correlation coefficient is

$$r = \frac{\Sigma\ (Xi - \overline{X})\ (Yi - \overline{Y})}{\sqrt{(\Sigma\ (Xi - \overline{X})^2\ \Sigma\ (Yi - \overline{Y})^2}}$$

where

Xi is the rank of the i-th X value

Yi is the rank of the i-th Y value

$\overline{X}$ is the mean of Xi

$\overline{Y}$ is the mean of Yi.

The formula shows that X and Y are interchangeable. So the degree of linear relationship between X and Y is also the degree of linear relationship between Y and X.

Expert Data Analysis and Comparison

Halstead's Metrics

C++ Analysis. As shown in Table III, Halstead's metrics correlate well with other Halstead's metrics. This result supports the study made by Li and Cheung [30] and Crawford, McIntosh, and Pregibon [17]. This finding suggests that

Halstead's metrics are variants of the size metrics. When Halstead's metrics are correlated with the other metrics as shown in Table III, they (except n_2) correlate well with McCabe's and Cox's sequence of operation metric. Halstead's unique operand count (n_2) performance is a little below the standard of the rest of the members.

Ada Analysis. Table IV shows the correlation coefficient among the selected metrics for the programs written by the experts in Ada. As in C++, Halstead's metrics also correlate well with other Halstead's metrics as well as with McCabe's metrics and Cox's sequence of operation metrics. The unique operand count also performs as well as the rest of the members. In Ada, Halstead's metrics seem to correlate well with Cox's data hiding. Table IV shows that Halstead's metrics and Cox's data abstraction are not linearly related. They also do not relate well to Henry and Kafura's data sharing metric.

Modula-2 Analysis. As in C++ and Ada, Halstead's metrics correlate well with other Halstead's metrics and McCabe's metric and correlate poorly with Cox's data abstraction metric. Analogously to Ada, Halstead's metrics correlate well with Cox's data hiding. When Halstead's metrics are correlated with Cox's sequence of operation, Modula-2's performance is not so good as in C++ or Ada. In Modula-2, Halstead's metrics show no linear relation with Henry and Kafura's data sharing metrics. Table V shows the

correlation coefficients for the Modula-2 analysis.

TABLE III

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR C++ PROGRAMS WRITTEN BY THE EXPERTS

	n1	n2	N1	N2	V	D	E	V(G)	H&K	TSO	DH	DA	N
n2	.96												
N1	.98	.90											
N2	.99	.92	.99										
V	.99	.95	.98	.99									
D	.98	.91	.99	.99	.99								
E	.99	.93	.97	.97	.98	.98							
V(G)	.95	.88	.96	.97	.96	.97	.94						
H&K	.84	.78	.87	.88	.86	.88	.80	.94					
TSO	.97	.93	.94	.94	.96	.95	.99	.90	.75				
DH	.90	.87	.91	.91	.91	.89	.83	.81	.76	.80			
DA	.27	.21	.35	.36	.32	.36	.14	.46	.71	.09	.38		
N	.98	.92	.99	.98	.91	.97	.95	.87	.89	.13	.92	.34	
Ne	99	.98	.97	.97	.94	.96	.98	.80	.83	.07	.89	.19	.97

C++ - Ada Analysis. When Halstead's metrics for C++ and Ada programs are correlated, they show a good linear relationship. This implies that the C++ program sizes increase at almost the same rate as the Ada program sizes. Halstead's metrics for C++ programs also correlate well with McCabe's metric and Cox's data hiding metric for Ada programs. Table VI shows the C++ - Ada comparison.

TABLE IV

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR ADA PROGRAMS WRITTEN BY THE EXPERTS

	n1	n2	N1	N2	V	D	E	V(G)	H&K	TSO	DH	DA	N
n2	.99												
N1	.99	.99											
N2	.99	.99	.99										
V	.99	.99	.99	.99									
D	.99	.99	.99	.99	.99								
E	.98	.99	.98	.99	.99	.98							
V(G)	.98	.97	.96	.97	.97	.97	.93						
H&K	.76	.74	.71	.73	.72	.75	.64	.85					
TSO	.98	.98	.97	.97	.98	.97	.99	.93	.65				
DH	.96	.96	.97	.97	.98	.97	.99	.89	.56	.98			
DA	-.38	-.40	-.31	-.34	-.35	-.35	-.37	-.41	-.43	-.41	-.26		
N	.99	.99	.99	.99	.99	.99	.99	.97	.72	.98	.97	-.11	
Ne	.99	.99	.99	.99	.99	.99	.99	.98	.75	.98	.96	-.17	.99

TABLE V

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR MODULA-2 PROGRAMS WRITTEN BY THE EXPERTS

	n1	n2	N1	N2	V	D	E	V(G)	H&K	TSO	DH	DA	N
n2	.99												
N1	.99	.99											
N2	.99	.99	.99										
V	.99	.99	.99	.99									
D	.99	.99	.99	.99	.99								
E	.99	.99	.99	.99	.99	.99							
V(G)	.97	.97	.97	.97	.97	.97	.95						
H&K	.58	.56	.61	.59	.57	.59	.50	.73					
TSO	.88	.89	.84	.86	.86	.86	.88	.85	.44				
DH	.98	.98	.97	.97	.98	.98	.97	.91	.44	.87			
DA	.31	.33	.35	.34	.33	.32	.27	.45	.59	.22	.30		
N	.99	.99	.99	.99	.99	.99	.99	.97	.60	.85	.97	.52	
Ne	.99	.99	.99	.99	.99	.99	.99	.97	.55	.88	.98	.49	.99

TABLE VI

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR C++ AND ADA PROGRAMS WRITTEN BY EXPERTS

(A represents Ada and C represents C++)

A\C	n1	n2	N1	N2	V	D	E	V(G)	H&K	TSO	DH	DA
n1	.98	.90	.98	.98	.97	.99	.99	.97	.87	.97	.83	.31
n2	.99	.92	.97	.98	.97	.99	.99	.97	.86	.98	.83	.29
N1	.97	.88	.98	.97	.96	.98	.99	.94	.83	.97	.83	.26
N2	.99	.92	.97	.98	.98	.99	.99	.97	.86	.97	.83	.27
V	.98	.90	.98	.98	.97	.99	.99	.95	.84	.98	.83	.25
D	.98	.89	.98	.98	.97	.99	.99	.96	.86	.97	.83	.29
E	.98	.93	.96	.96	.97	.94	.99	.92	.78	.99	.81	.15
V(G)	.94	.85	.96	.96	.95	.97	.95	.99	.93	.91	.78	.45
H&K	.70	.60	.75	.76	.73	.77	.67	.85	.96	.60	.61	.79
TSO	.96	.91	.94	.94	.95	.95	.99	.92	.77	.99	.75	.11
DH	.97	.91	.95	.94	.95	.95	.99	.88	.71	.99	.83	.06
DA	-.29	-.31	-.20	-.22	-.27	-.26	-.34	-.36	-.40	-.37	.04	-.21

Table IX shows a listing of the mean values for the C++, Ada, and Modula 2 complexity metrics. According to the table, Ada programs on the average are slightly larger than C++ programs. The Ada experts seem to use more operands. This could be because of the large size of the syntactic constructs of the Ada language. For this reason, the Ada programs are estimated to have a larger program length than C++ programs. Halstead suggested that the difficulty increases with more operators and decreases with more operands [14]. As shown in Table IX, according to Halstead's metric in this study, the Ada programs are less difficult to

write than the C++ programs. It is shown that in this study writing Ada programs requires more effort than writing C++ programs. The effort increases with the difficulty and program volume (E = V * D).

C++ - Modula-2 Analysis. Similar to C++ and Ada, C++ and Modula-2 Halstead's metrics show a strong linear relationship. Halstead's metrics for C++ programs also correlate well with McCabe's metric and Cox's data hiding metric for Ada programs. Table VII shows this relationship. When the averages of C++ and Modula-2 Halstead's metrics are compared, Table IX shows that C++ programs consist of more operators and fewer operands than Modula-2 programs. This causes the average of Halstead's difficulty metric to be higher for C++ programs than for Modula-2 programs. Table IX also shows that, on the average, a Modula-2 program length is larger than a C++ program length.

Ada - Modula-2 Analysis. As shown in Table VIII, the Ada - Modula-2 analysis follows the same pattern as the C++ - Modula-2 analysis. Halstead's metrics of Ada programs have a strong linear relationship to Halstead's metrics, McCabe's metric, and Cox's data hiding metric of Modula-2 programs. On the average, Modula-2 programs in this study are larger than Ada programs which cause higher program volume. The Modula-2 programs consist of fewer unique operands which result in higher difficulty than the Ada programs. Higher program volume and higher difficulty in turn result in

Modula-2 programs having a higher effort metric than the Ada programs.

TABLE VII

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR C++ AND MODULA-2 PROGRAMS WRITTEN BY EXPERTS

(C represents C++ and M represents Modula-2)

M\C	n1	n2	N1	N2	V	D	E	V(G)	H&K	TSO	DH	DA
n1	.99	.93	.97	.97	.98	.98	.99	.95	.82	.99	.84	.22
n2	.99	.94	.97	.98	.98	.98	.99	.95	.82	.99	.83	.20
N1	.99	.92	.98	.98	.98	.99	.99	.95	.83	.98	.86	.26
N2	.99	.93	.98	.98	.98	.98	.99	.95	.83	.99	.85	.23
V	.99	.93	.98	.98	.98	.98	.99	.94	.82	.99	.84	.21
D	.99	.93	.98	.98	.98	.98	.99	.95	.83	.99	.84	.24
E	.98	.93	.95	.95	.96	.96	.99	.92	.77	.99	.81	.12
V(G)	.96	.88	.97	.97	.96	.98	.96	.99	.91	.94	.80	.39
H&K	.59	.46	.68	.68	.63	.69	.56	.77	.91	.47	.55	.88
TSO	.88	.93	.78	.81	.86	.81	.86	.84	.74	.88	.67	.18
DH	.98	.96	.95	.95	.96	.95	.98	.89	.74	.99	.87	.10
DA	.39	.35	.49	.51	.45	.47	.33	.54	.57	.25	.51	.52

McCabe's Metric

C++ Analysis. As stated above, McCabe's metric correlates well with Halstead's metrics. This supports the assumption that the larger a program is, the more condition statements might come out in the program. It also correlates really well with Henry and Kafura's data sharing as shown in Table III. This is surprising because the only thing they

have in common is that they have linear relationships with Halstead's metrics. McCabe's metric is basically the number of conditions in a program while Henry and Kafura's data sharing metric is the average of the amount of data passed multiplied by the amount of data received. As expected, McCabe's metric correlates well with Cox's step of operations metric.

TABLE VIII

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR ADA AND MODULA-2 PROGRAMS WRITTEN BY EXPERTS

(A REPRESENTS ADA AND M REPRESENTS MODULA-2)

M\A	n1	n2	N1	N2	V	D	E	V(G)	H&K	TSO	DH	DA
n1	.99	.99	.99	.99	.99	.99	.99	.95	.69	.99	.98	-.36
n2	.99	.99	.99	.99	.99	.99	.99	.96	.70	.98	.98	-.32
N1	.99	.99	.99	.99	.99	.99	.99	.96	.70	.98	.98	-.32
N2	.99	.99	.99	.99	.99	.99	.99	.96	.69	.98	.98	-.34
V	.99	.99	.99	.99	.99	.99	.99	.95	.68	.99	.98	-.34
D	.99	.99	.99	.99	.99	.99	.99	.96	.70	.98	.98	-.36
E	.98	.99	.98	.98	.99	.98	.99	.92	.62	.99	.99	-.37
V(G)	.98	.99	.97	.98	.98	.98	.96	.99	.80	.95	.91	-.39
H&K	.66	.64	.63	.64	.62	.66	.52	.78	.96	.52	.44	-.33
TSO	.84	.88	.79	.82	.84	.81	.88	.82	.59	.88	.82	-.56
DH	.95	.96	.95	.96	.97	.95	.98	.89	.56	.96	.99	-.26
DA	.37	.36	.35	.36	.34	.36	.28	.45	.55	.27	.26	.40

TABLE IX

MEANS AMONG THE SELECTED METRICS FOR
THE C++ - ADA - MODULA-2 COMPARISON

	C++	Ada	Modula-2
n1	70	73	69
n2	49	64	53
N1	288	273	285
N2	493	494	522
V	4130	4186	4291
D	394	294	346
E	3529528	4115969	4991232
L	0.09	0.22	0.10
V(G)	15	17	13
H&K	17	29	47
TSO	1	3	3
DH	10	11	10
DA	3	2	2
N	781	767	807
Ne	488	579	502

Ada Analysis. Besides correlating well with Halstead's metrics in Ada programs, McCabe's metric also correlates well with Cox's step of operations metric. McCabe's metric in Ada programs does not correlate well with Henry and Kafura's metric or with Cox's step of operations metric.

Modula-2 Analysis. In Modula-2, McCabe's metric correlates well not only with Halstead's metric but also with Cox's data hiding metric. The latter outcome is surprising because they seem to have no linear relationship. Since no research is known to make a comparison of this kind, it is difficult to argue about the relationship between McCabe's and Cox's data hiding metric. It is

interesting to find out whether or not the correlation coefficient between the two metrics will be as good when more programs are observed.

C++ - Ada Analysis. When McCabe's metric of C++ programs is compared to the metrics of Ada, the following observations can be made. McCabe's metric of C++ programs correlates well with Halstead's metrics, McCabe's metric, and Cox's step of operations of Ada. On the average, McCabe's metric of C++ programs is smaller than McCabe's metric of Ada programs.

C++ - Modula-2 Analysis. McCabe's metric of C++ programs correlates well with Halstead's metrics and McCabe's metric of Modula-2 programs. When McCabe's metric of C++ and Modula-2 are compared, on the average C++ programs have more condition statements than Modula-2 programs.

Ada - Modula-2 Analysis. Analagous to the C++ - Modula-2 analysis, McCabe's metric of Ada correlates well only with Halstead's and McCabe's metrics of Modula-2. On the average, Ada programs consist of more condition statements than Modula-2 programs.

Henry and Kafura's Data Sharing Metric

In this study, the data sharing metric of C++ programs correlates well with McCabe's metric of C++, Ada, and

Modula-2 programs. From this fact, it should be concluded that the data sharing metric of C++ depends on the size of a program. On the contrary, the data sharing metric of C++ does not correlate well with Halstead's metrics. As expected, the data sharing metric of C++ programs is linearly related to the data sharing metric of Ada and Modula-2 programs. When the data sharing metric of Ada programs is correlated with metrics of Modula-2 programs. it correlates well only with the data sharing metric of Modula-2 programs.

As discussed in the previous chapter, Henry and Kafura's data hiding metric increases with the increase of shared data in a program. When the average data hiding metric among C++, Ada, and Modula-2 programs are compared, it is found that C++ programs have the least shared data followed by Ada and then Modula-2. One question that can be raised from this finding is "Does lower data sharing correlate well with the size metric?". From the formula of Henry and Kafura's data sharing metric, FanIn * FanOut, the answer should be "yes" if the number of modules metric correlates well with the size metric and "no" otherwise. FanIn for a module is the number of modules that pass data to the module either directly or indirectly and FanOut for a module is the number of modules to which data is passed either directly or indirecly [14]. If the shared data are eliminated, basically the formula becomes M * M where M is the number of modules.

Cox's Surface Area Metrics

Time Sequence of Operations Metric (TSO). The TSO metric of C++ programs correlates well with Halstead's and McCabe's metrics of C++, Ada and Modula-2 programs. When the TSO metric of Ada is correlated with metrics of Modula-2 programs, only Halstead's and McCabe's metrics of Modula-2 programs correlate well with TSO metrics of Ada programs. The step of operations metric by definition is the number of steps that a user needs in order to know how to use an object. In object-oriented style, when a program is getting larger, it is most likely that the number of objects increases. According to Cox [16], the more sequence of operations a user needs in order to know how to use an object, the more complex the object is. In this study, among C++, Ada, and Modula-2, objects written in C++ require the least number of sequence of operations while objects written in Ada and Modula-2 require almost the same number of sequence of operations.

Data Hiding Metric. In C++, the data hiding metric correlates well only with Halstead's metrics, especially the program volume. The data sharing metric of C++ has no linear relationship to any of the Ada and Modula-2 metrics. In Ada, the data hiding metric correlates well with Halstead's, McCabe's, and TSO metrics. The data hiding metric of Ada also correlates well with data hiding, Halstead's, and McCabe's of Modula-2. In Modula-2, the data

hiding metric is linearly related to Halstead's, McCabe's, and TSO metrics. When the averages of the data hiding metrics of Ada, C++, and Modula-2 are compared, they all have almost the same ordinal number.

Data Abstraction Metric. In all three languages, Ada, C++, and Modula-2, there is no linear relationship between the data abstraction metric and the rest of the metrics. As with the data hiding metric, the average data abstraction metric of the three languages is almost the same. This suggests that the data abstraction metric does not depend on the size of a program. It also suggests that the expert programmers tended to create the same amount of data abstraction in Ada, C++, and Modula-2 when object-oriented style was applied.

Experts' Data Analysis Overview

In this study, Halstead's metrics of the expert programs seem to correlate well with McCabe's metric. This is also true when Halstead's metrics of C++ programs are correlated with Halstead's and McCabe's metrics of Ada or Modula-2 programs. Thus, we can infer, in this study, that McCabe's metric increases as the size of the program grows larger. The only exception is the unique operand count metric (n_2). This is understandable because the number of operands in a language is fixed but program sizes are varied. From the difficulty metric formula, Halstead hypothesized that it is easier to write a program in a

language with a large number of operands than in a language with few operands. Table IX shows that, on the average, Ada programs have the smallest number followed by Modula-2 then C++ programs in the difficulty metric.

Henry and Kafura's data sharing metric of Ada, Modula-2, and C++ programs have good linear relationships. This observation suggests that the data sharing metric increases at the same rate in Ada, Modula-2, or C++ programs. It is found in this study that, on the average, C++ programs consist of the least data shared followed by Ada and then Modula-2 programs. In all three languages, the data sharing metric does not correlate well with Halstead's metric. This suggests that the data sharing metric does not depend on the program size.

Cox's surface area metric is used to exploit the advantage of the object-oriented technique. In general, TSO correlates well with Halstead's and McCabe's metrics. When the data hiding metric is observed, it correlates well with Halstead's, McCabe's, and TSO metric for Ada and Modula-2 programs and only to Halstead's metrics for C++ programs. The data abstraction metric is the only metric which does not correlate well with the rest of the metrics. This is understandable because this metric is merely an interface metric for the number of distinct arguments in the functions used for an object [16]. Especially in surface area metrics, C++ programs do not correlate well with Ada and Modula-2 programs. On the contrary, the surface area metrics of Ada

programs correlate well with Modula-2 programs. This is probably because of the way objects are defined in Ada, C++, and Modula-2. Ada and Modula-2 experts defined an object in almost similar encapsulated modules called module in Modula-2 and package in Ada while C++ experts use a user data type called class to define an object.

There are some factors that need to be discussed in order to avoid misunderstanding the results of this experiment, especially concerning the observations made about the programs written by the experts. First of all, all of the programs are written in the object-oriented style so the result does not generalize to other styles of programming. Second, all of the experts' programs are selected from various textbooks. The experts are the authors of the books. So the data observed are not obtained from a controlled experiment. Third, the average program size is about 750 tokens. So, a similar study with a larger program size to observe might not obtain a similar result. Table X shows each program's size in lines of codes and tokens. Finally, because of the difficulty in finding a similar program written in Ada, C++, and Modula-2, only eight programs for each language are observed. Again, the results of this study might not reflect the results of a similar study with a larger number of programs. With all of these biasing factors, the data are still worth analyzing and show the same results when compared with the idealized experiment (when all of the biasing factors are removed).

TABLE X

EXPERT PROGRAMS AND THEIR SIZES IN LINES OF CODES AND TOKENS

Program		LOC	Tokens
Table	(a)	201	1009
	(b)	182	933
	(c)	164	832
Complex	(a)	66	356
	(b)	73	449
	(c)	61	404
Matrix	(a)	37	271
	(b)	43	281
	(c)	46	279
Rational	(a)	162	790
	(b)	71	483
	(c)	86	520
List	(a)	80	412
	(b)	60	326
	(c)	59	297
Stack	(a)	88	474
	(b)	32	163
	(c)	79	322
Dictionary	(a)	479	2469
	(b)	726	3274
	(c)	826	3365
Queue	(a)	88	470
	(b)	38	226
	(c)	101	452

(a) written in C++

(b) written in Ada

(c) written in Modula-2

Novice Data Analysis and Comparison

As discussed in chapter IV, the data gathered from novice programmers were produced under time constraints. This caused more than half of the targeted programs to be unfinished. Because of the quantity of the unfinished (partial) programs, it is impossible to void them and still have a reasonable analysis. Partial programs either could be syntactically unfinished programs, or they could be programs with compilation or run-time errors.

Reynolds [38] discusses metrics to measure the complexity of partial programs. His discussion of partial programs is the context of the stepwise refinement process, that is, "partial" meaning not yet fully refined into code in the software development life cycle.

> "In order to monitor stepwise refinement of pseudocode module, they (the metrics) must be extended to describe the complexity of a partially completed program. Such a pseudocode program contains two classes of symbolic terms. The first class of terms, called prescribed terms, corresponds to reserved words and symbols in the target language. The other class of terms stands for inferences about aspects of the program that remain to be instantiated. These are called projected terms." [38]

A slight adaptation of these concepts is utilized to

measure the complexity of partial (i.e., incompletely written) programs in this work. Basically, Reynolds claims that a complete program is composed of prescribed and projected components. The prescribed components are the partial operators and operands in the case of Halstead's metrics preexisting in the incompletely developed program. The projected components are the tokens that should be contained in a complete program. The component can be represented by <S> for statements that have two operands and one operator, <E> for expressions that have one operator and one operand, <O> for operands, or < > for operators.

The projected components can consist of only <S>s (the crude estimation) or consist of only <O>s and < >s (the best estimation). For example, if the projected components consist of ten statements, the primitive projected operands will be twenty and the projected operators will be ten. The more <S>s and <E>s represented by <O>s and < >s, the closer is the measurement to the perfect estimation.

The method outlined above is not directly applicable for the analysis of incomplete or partial programs in this study, since the method clearly is not used for comparison purposes but to provide a quantitative description of the program development process [38]. One possible way to adapt this method for comparison is to let the novice programmers finish their programs by estimating the projected operands and operators. This is a plausible adaptation. Nevertheless, the result will be arguably biased because of different

estimation and finishing times.

Two methods were developed to analyze the partial programs. They are called extrapolation and percentage methods. These methods basically assume that the measured complexity metrics are linearly related to the size of the programs. In the extrapolation method, the following formula is used to estimate the complexity metrics.

$$\frac{N_{partial}}{N_{estimated}} = \frac{M_{partial}}{M_{estimated}}$$

Where

$N_{partial}$ is the length of the partial program which is the sum of the total number of operators and operands, $N_1 + N_2$;

$N_{estimated}$ is the "estimated" complete length which is the corresponding expert program in terms of the operator and operand tokens;

$M_{partial}$ is the complexity metric of the partial program;

$M_{estimated}$ is the complexity metric that needs to be estimated.

The above formula can be rewritten as

$$M_{estimated} = \frac{M_{prescribed}}{N_{prescribed}} N_{estimated} \cdot [n]$$

For example, to estimate the difficulty metric of a partial program the following formula is used.

$$D_{estimated} = \frac{D_{partial}}{N_{partial}} N_{estimated}$$

As shown in the above formula [n], the extrapolation method depends heavily on the length metric. The estimated Halstead's or McCabe's metrics would also provide meaningful extrapolations to the complete programs but the estimated Henry and Kafura's or Cox's surface area metrics may not be good indicators of the complete programs due to the weak correlation between the length metric and Henry and Kafura's metric or Cox's surface area metric.

The percentage method basically measures the percentage of the length of the partial programs over the length of the estimated programs. The formula of the percentage method is

$$\% \text{ of } N_{partial} = \frac{N_{partial}}{N_{estimated}} \times 100\% \quad [m].$$

Undoubtedly, an argument for the use of the extrapolation and percentage methods may be made due to their usefulness in comparing partial programs. It is not the purpose of this study to develop a metric to measure partial programs. As indicated before, the work of Reynolds [38, 39] is not directly applicable to this study. Even though the extrapolation and the percentage methods are arguable, the methods merit consistency which eliminates biasing factors such as the bias introduced by using the estimated length. Since all of the estimated metrics are extrapolated with respect to the estimated length, when they are compared, the estimated length will not be the important

factor.

As mentioned in chapter IV, novice programmers are divided into two groups. Chapter III includes a discussion of the reasons for the design which are for randomizing and minimizing the biasing factors such as the learning curve. When analyzing the data, the groups will be combined and analyzed as Ada and C++ data as shown in Tables XI and XII.

In the following discussion, first some observations on the subjects and their opinions about the experiment are discussed. Then the analysis will be presented with the percentage method followed by the extrapolation method.

Since the number of subjects involved in the novice experiment was only eleven, their activities were observed closely. On the first day of the experiment, group 1, consisting of six members, who implemented the problems in Ada language, seemed to have no difficulty in writing the programs. Table XII shows that all members of group 1 finished at least one of the three problems assigned. Three members finished the second problems. Most of their time was spent in debugging the programs. However, group 2, consisting of five members, had a difficult time implementing the problems in C++. No one completed a single program of the three problems assigned. They basically did not understand the concept of class in C++. They saw a class as a structure type in C rather than a data abstraction tool.

On the second day of the experiment, only one problem

was given to the subjects. The difficulty of the problem was supposed to be equal to the three problems assigned on the first day. Group 1 implemented the problem in C++ while group 2 implemented the problem in Ada. Of the six members, only two of them completed the programs but failed to run them successfully. When their programs were examined closely, minor errors were detected. But, the more disturbing factors about the programs were the way the subjects treated the class structure. They entered unnecessary data into the class structure. Again, they did not see the class structure as a data abstraction tool. In group 2, most members finished the program but had difficulty compiling it. As in group 1, they did not seem to understand the concept of data abstraction and encapsulation. They used them without knowing the advantage of the package in Ada.

At the end of the experiment, the subjects were given the post-experiment questionnaire (Appendix D). The following discussion summarizes their opinions about the experiment. The majority felt that Ada was easier to learn and use than C++. This, of course, made them prefer Ada as their favored programming language over C++. The only exception was the subjects who had programmed in C for more than three semesters. They prefered C++ over Ada even though they still thought that Ada was easier to learn and use. Most of them felt that they needed more time to learn a new language.

It is possible that if the subjects were given the same tasks and asked to write them in the programming style they were accustomed to, such as top-down design, they would have finished the tasks. In this experiment, the subjects were forced to write the programs in the object-oriented style by providing them with a fixed main driver (Appendix B and Appendix C) to test their objects. Many factors can be blamed for inadequately finished programs by the subjects but the most significant factor is believed to be the treatment. Considering the level of knowledge of the subjects, who were mostly seniors and graduate students, early in the experiment it was assumed that three hours of treatment to learn basic object-oriented style were sufficient. The reasoning behind this number of hours was that the more treatment the subjects got the more likely the biasing factor such as an uneven learning curve would arise. Even though their programs were not ideal object-oriented programs, they could be considered as programs written with the object-oriented programming paradigm in mind.

Percentage Method

As shown in the percentage formula above [m], this method measures the percentage of the novices' program length compared to the experts' program length. This measurement implies the percentage of the partial programs compared to the finished program. Tables XI and XII show the percentage of the partial programs over the completed

programs written in C++ and Ada, respectively. Table XI indicates that none of the novice programmers could finish a program in C++ as required. In some instances, they completed the programs but were not able to fix the compilation or run-time errors. Only three of the eleven novice programmers finished over seventy-five percents and the rest finished below fifty percent of the final program required. Table XII shows that six of the novice programmers finished at least one program, three finished two programs, and five finished below fifty percent of the total number of programs.

TABLE XI

PERCENTAGE OF NOVICE PROGRAMS' LENGTH OVER EXPERT PROGRAMS' LENGTH IN C++

Program	Programmer	N	Ne	Completed
Table	1	24%	49%	No
	2	32%	73%	No
	3	27%	59%	No
	4	20%	40%	No
	5	21%	40%	No
Complex	4	79%	70%	No
Matrix	7	40%	106%	No
	8	93%	170%	No
	9	41%	117%	No
	10	136%	186%	No

TABLE XII

PERCENTAGE OF NOVICE PROGRAMS' LENGTH
OVER EXPERT PROGRAMS' LENGTH IN ADA

Program	Programmer	N	Ne	Completed
Table	7	34%	24%	No
	8	23%	22%	No
	9	24%	18%	No
Complex	1	60%	100%	Yes
	2	34%	70%	No
	3	59%	84%	No
	4	75%	100%	Yes
	5	72%	84%	Yes
Matrix	1	141%	112%	Yes
	2	112%	102%	Yes
	3	102%	105%	Yes
	4	111%	121%	Yes
	5	98%	93%	Yes
	6	108%	108%	Yes
List	1	21%	41%	No
	2	62%	60%	No

In Tables XI and XII, N and $N_{estimate}$ represent Halstead's program length and Halstead's estimated program length. The Table shows that the percentage of $N_{estimate}$ doubles the percentage of N when the percentage of N is low. When the percentage of N is high, as indicated in the completed programs of Table XII, the percentage of $N_{estimate}$ approaches an equilibrium. This does not mean that the N and $N_{estimate}$ metrics are almost equal for the complete programs. It just suggests that the ratio of the partial

program length over the completed program length is equal to the ratio of the estimated one.

Extrapolation Method

Table XIII shows Halstead's and McCabe's metrics for the novice programs written in Ada after the partial metrics were extrapolated by Halstead's length metric as discussed above. In this table, Halstead's metrics do not correlate well with their own member metrics or with McCabe's metric except in a few instances. When the partial metrics are extrapolated by Halstead's estimate length as shown in Table XIV, the correlation coefficients are better except for volume and effort metrics. However, for C++, as shown in Tables XV and XVI, Halstead's metrics have good correlation coefficients between their own members but not with McCabe's metric. Correlation coefficients in Tables XV and XVI are quite similar. As indicated in the caption, Table XV contains the correlation coefficients among Halstead's and McCabe's metrics for the novice programs written in C++ after extrapolation by Halstead's program length. Table XV is the same as Table XVI except for the fact that it is extrapolated by Halstead's estimated program length.

When the metrics extrapolated by program length are correlated with the same metrics extrapolated by estimated program length as shown in Table XVII, they correlate well. This indicates that program length and estimated program length have a good linear relationship

which is also shown in Table XIII through XVI.

TABLE XIII

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR NOVICES' ADA PROGRAMS AFTER EXTRAPOLATION BY N

	n1	n2	N1	N2	V	D	E	V(G)	N
n2	.93								
N1	.07	.06							
N2	.87	.89	.12						
V	.18	.13	-.01	.20					
D	.86	.73	.14	.91	.26				
E	.61	.46	.16	.78	.21	.93			
V(G)	.73	.79	.24	.89	.03	.74	.63		
N	.88	.88	.13	.99	.24	.93	.80	.86	
Ne	.98	.98	.09	.93	.17	.85	.61	.81	.93

TABLE XIV

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR NOVICES'S ADA PROGRAMS AFTER EXTRAPOLATION BY Ne

	n1	n2	N1	N2	V	D	E	V(G)	N
n2	.96								
N1	.96	.95							
N2	.96	.97	.99						
V	.41	.38	.45	.41					
D	.96	.89	.97	.96	.43				
E	.86	.75	.91	.88	.39	.96			
V(G)	.93	.95	.95	.97	.34	.91	.82		
N	.96	.96	.99	.99	.43	.97	.89	.97	
Ne	.99	.99	.97	.98	.41	.95	.83	.95	.98

TABLE XV

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR NOVICES' C++ PROGRAMS AFTER EXTRAPOLATION BY N

	n1	n2	N1	N2	V	D	E	V(G)	N
n2	.91								
N1	.97	.88							
N2	.98	.92	.99						
V	.98	.88	.99	.99					
D	.97	.79	.95	.95	.97				
E	.86	.59	.82	.81	.86	.94			
V(G)	.83	.87	.82	.87	.82	.75	.55		
N	.98	.91	.99	.99	.99	.95	.82	.85	
Ne	.99	.93	.96	.98	.98	.96	.84	.82	.97

TABLE XVI

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR NOVICES' C++ PROGRAMS AFTER EXTRAPOLATION BY Ne

	n1	n2	N1	N2	V	D	E	V(G)	N
n2	.91								
N1	.88	.86							
N2	.94	.92	.97						
V	.94	.89	.98	.99					
D	.97	.79	.88	.92	.95				
E	.82	.55	.73	.76	.81	.93			
V(G)	.80	.86	.65	.77	.72	.70	.46		
N	.92	.90	.99	.99	.99	.91	.75	.73	
Ne	.99	.92	.88	.94	.94	.95	.80	.80	.92

Table XVIII shows the average values of the selected metrics after extrapolation by Halstead's program length and Halstead's estimated program length. It shows that in C++, the average of Halstead's metrics after extrapolation by program length are almost double the metrics after extrapolation by estimated program length. While in the Ada counterpart, the average values are almost the same. This may suggest that the ordinal values in Table XVIII are better estimated when applied to programs written in Ada than written in C++ by novice programmers. On the other hand, the difference between the pairs of values in C++ columns may have been caused by the fact that many of the programs written in C++ are half finished. The big adjustment may have caused the big difference in the metrics after extrapolation by program length and estimated program length.

Because of the bad correlation of the program level metric, the metric is compared by averaging as done by Conte, Dunsmore, and Shen [14]. As shown in Table XVIII, it is suggested that for novice programmers C++ is a more complex language than Ada. However, it needs to be pointed out that studies have shown that the language level is a decreasing function of program size [14]. So the language level is not a constant number and in a manner analogous to other Halstead's metrics, it depends on the program size.

Table XVIII also shows the comparison of Henry and Kafura's metric and Cox's surface area metrics between C++ and Ada. The correlation coefficient is not used to analyze data because the correlation coefficients produced do not correlate well with the rest of the metrics. On the average, Henry and Kafura's metric for C++ programs is larger than the metric for Ada programs. This observation suggests that novice programmers tend to pass and receive more data between modules or use more external variables (implicit passing and receiving between modules) in Ada programs than in C++ programs.

The average Cox's time sequence of operations metric (TSO) for C++ programs is less than the average of the same metric for Ada programs. This is basically caused by the way the languages define an object in object-oriented programs. The C++ language can automatically create the data object when an object is initiated and also automatically destroy it when the object is not in use. Ada, however, has to create the data object before the object can be used properly and the programmer has to explicitly destroy the object.

When the averages of Cox's data hiding metric for C++ and Ada are compared, Ada programs have more ordinal numbers than C++ programs. As discussed in the previous chapter, the data hiding metric is the sum of all variables, procedures, functions, and data types inside the specification part of an object. Considering the definition

of the data hiding metric, the differences among the four ordinal numbers are not substantial. So it can be assumed that the novice programmers use almost the same amount of non-private data in creating an object in Ada and C++ programs.

Cox's data abstraction metric is defined as the number of data types required to understand how to use an object. As in the data hiding metric, the difference between the data abstraction metric of C++ and Ada programs is not sunstantial enough to be considered different. It can be said that in this experiment, the novice programmers used as many data types in Ada programs as in C++ programs in creating an object.

TABLE XVII

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR NOVICES' PROGRAMS AFTER EXTRAPOLATION BY N AND Ne

	Ada	C++
n1	.89	.97
n2	.91	.92
N1	.98	.86
N2	.96	.91
V	.95	.92
D	.94	.97
E	.83	.98
L	.83	.94
V(G)	.99	.99
N	.95	.90
Ne	.92	.97

TABLE XVIII

MEANS AMONG THE SELECTED NOVICES' METRICS
FOR THE C++ - ADA COMPARISON AFTER
EXTRAPOLATION BY N AND Ne

	C++ (N)	C++ (Ne)	C++	Ada (N)	Ada (Ne)	Ada
n1	98	49		42	41	
n2	71	35		46	46	
N1	245	29		234	173	
N2	404	209		289	297	
V	2656	1379		2724	2482	
D	294	149		135	136	
E	312338	159509		141034	146467	
V(G)	19	9		10	12	
H&K			15			25
TSO			1			2
DH			31			35
DA			4			5
N	649	338		461	469	
Ne	579	286		274	274	
L	.45	.21		.54	.46	

(N) the values are extrapolated with program length
(Ne) the values are extrapolated with estimated program length

Novices' Data Analysis Overview

Unlike the expert programmers, the novice programmers developed their programs under a controlled experiment. Problems emerged when more than half of the programs produced by the novices were unfinished. Two methods were

devised to measure the novices' programs in order to overcome the problem. They are called the percentage and extrapolation methods. The percentage method measures the percentage of the length of unfinished programs over the finished programs. The extrapolation method estimates the metrics of a finished program from the partial program.

In this section, the percentage of the finished Ada and C++ programs were compared and discussed. Also included in this section is the analysis of Halstead's, McCabe's, Henry and Kafura's, and Cox's surface area metrics of Ada and C++ programs after the two extrapolation methods were applied. To get the estimated metrics of partial programs, the partial metrics are extrapolated using Halstead's program length factor and Halstead's estimated program length factor.

One question that can be raised for this experiment is "What language is better suited for the novice programmers to learn the object-oriented technique?". According to Tables XI and XII, the answer has to be "Ada". More programs are completed in Ada than in C++. When the same question was asked directly of the novice programmers, most of them felt that Ada was much easier to learn and use than C++. When Table XVIII is observed, it is evident that C++ programs are more complex than Ada programs in terms of Halstead's and McCabe's metrics and less complex in terms of Henry and Kafura's and Cox's surface area metrics.

As in the experts' analysis, it is important to state

the factors and limitations of the outcome. All of the observations of the novices' data are limited by the following factors. First, the data analyzed are the extrapolated data. Even though it is the purpose of the experiment to analyze complete programs, the result might not reflect similar experiments that measure the programs directly without extrapolation methods applied. Secondly, the programs produced are under a controlled experiment specified in the previous chapter. A different controlled experiment might not have the same result. Finally, the average of the program size is 650 tokens for C++ where ten programs are observed and 450 tokens for Ada where sixteen programs are observed. So, a similar experiment with larger program sizes and a larger number of observations might not reflect the analysis above.

As discussed in Conte, Dunsmore, and Shen [14], the ideal controlled experiment is costly, time consuming, and probably impossible. This experiment was an attempt to meet the requirements necessary to have a valid result according to Conte, Dunsmore, and Shen. It is true that the outcome might not reflect as wide of a population as expected but the outcome is still worth analyzing and comparing to other similar studies.

Expert and Novice Data Analysis and Comparison

In the previous sections experts' and novices' programs were analyzed and compared. The fact that the

experts' programs and the novices' programs were obtained in different ways was also discussed. The experts' programs were obtained from the textbook which the experts wrote and the novices' programs were obtained from a controlled experiment. Clearly, the experts' and the novices' programs cannot be compared on a one to one basis. But, if the experts' programs are seen as the completed programs, the experts' and the novices' programs could be compared by focusing on how close the novices' programs are to the experts' programs.

The experts' and the novices' programs are analyzed by using correlation coefficients applied to Halstead's, McCabe's, Henry and Kafura's, and Cox's metrics. Due to the differences in the number of subjects and programs between the expert and novice groups, some experts' data needed to be duplicated and extracted. There are eight programs for the eight problems produced by the expert group in each language (Ada and C++). However, the novices were assigned only four of the eight problems. To make the metrics comparable, the experts' programs that the novices did not produce are ignored.

Since five novice programmers solved a similar problem, the metrics produced by the expert for that program were replicated as often as necessary to match the number of novices who wrote that program. Table XXIII illustrates the adjustment made for the expert versus novice comparison.

After the adjustment (duplication and extraction),

there are ten novices' and experts' C++ programs as well as sixteen novices' and experts' Ada programs to be observed. As discussed in the novice analysis above, there are two sets of novice metrics. One is produced by extrapolating the selected metrics with Halstead's program length and the other one is produced by extrapolating the selected metrics with Halstead's estimated program length. In the analysis which follows, both sets of metrics are compared with the expert metrics. The correlation coefficient tables for the novices' and experts' comparison (Table XIX and Table XX) are different compared to the ones in the novice analysis or expert analysis. The tables show only the correlation coefficients of the experts' metrics correlated to the similar novices' metrics. The correlation between different metrics is irrelevant for the analysis. The novices' and experts' comparison for Ada programs is presented in the next section followed by the novices' and experts' comparison for C++ programs.

Experts' and Novices' Comparison for Ada Programs

Table XIX shows that expert metrics correlate better with the novice metrics that have been extrapolated with Halstead's estimated program length. This finding supports the correlation coefficients in the novice analysis as shown in Table XIII and XIV which also suggested that the novice metrics extrapolated by estimated program length correlate better with the other metrics than the metrics extrapolated

by program length.

As shown in Table XIX, Halstead's and McCabe's expert metrics correlate well with the corresponding novice metrics with the exceptions of Halstead's number of operators metric and the volume metric. The good correlation coefficient between the experts' and novices' metrics was surprising because the novices' programs are mostly partial programs. The extrapolation of the novice metrics was believed to be the reason for this rather unexpected result.

TABLE XIX

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR NOVICES' AND EXPERTS' COMPARISON FOR ADA PROGRAMS

	Expert vs Novice(N)	Expert vs Novices(Ne)
n1	.88	.99
n2	.86	.98
N1	.11	.90
N2	.99	.96
V	.20	.46
D	.93	.93
E	.77	.85
V(G)	.99	.99
N	1.00	.95
Ne	.92	1.00

(N) the values are extrapolated with program length.

(Ne) the values are extrapolated with estimated program length.

Table XXI shows the average value of the experts' and the novices' metrics. As shown in the table, Halstead's and McCabe's metric between the novices and the experts are very close to each other. On the average, Henry and Kafura's expert metric is also close to the similar novice metric. The experts seemed to have more operations in using an object than the novices (refer to the TSO metric). This is partly caused by a good programming practice followed by the experts which is always destroying an object when it is not needed. Cox's data sharing and data abstraction metrics in Table XXI show that the novices used more unprotected data than the experts. This result is expected since a lot of the novice Ada programmers defined data in the specification part of the Ada package for easy access to the data while the experts defined them in the package body or declared them as private. Table XXI shows that the experts' Ada program level is more complex than the novices' Ada program. This proves again that Halstead's program level is not a fixed value as Halstead had expected but simply a decreasing function of the size metrics.

Novices' and Experts' Comparison for C++ Programs

When the experts' metrics for C++ programs are correlated to the corresponding novices' metrics, they produce a high correlation coefficient. As shown in Table XX, this is true for Halstead's and McCabe's metrics. Unlike

the Ada comparison, the correlation coefficients between the expert metrics and the novice metrics extrapolated by the program length are almost similar to the correlation coefficients between the expert metrics and the novice metrics extrapolated by the estimated program length. This finding supports the assumption stated in the previous section that the extrapolation of the novice data causes the novices' metrics to have a good correlation with the corresponding experts' metrics.

Table XXII shows the average value among the selected metrics for the novices' and experts! programs written in C++. It shows that, on the average, the novice programmers used more unique operands and unique operators than the expert programmers even though the novices' and the experts' program sizes are almost the same. This finding is also surprising since it is expected that the expert programmers use more unique operands than the novice programmmers. Since the novice programmers used more unique operands, according to Halstead, their programs need less effort to write than the expert programs. The program level of the experts and the novices is also different. The average program level value only indicates that the experts' programs are more complex and difficult to write than the novices' programs.

TABLE XX

CORRELATION COEFFICIENTS AMONG THE SELECTED METRICS FOR THE NOVICES' AND EXPERTS' COMPARISON FOR C++ PROGRAMS

	Expert vs Novice(N)	Expert vs Novices(Ne)
n1	.97	.99
n2	.90	.92
N1	.99	.88
N2	.99	.92
V	.99	.93
D	.95	.95
E	.82	.79
V(G)	.87	.83
N	1.00	.90
Ne	.97	1.00

TABLE XXI

MEANS AMONG THE SELECTED METRICS FOR NOVICES' AND EXPERTS' COMPARISON FOR ADA PROGRAMS

	Expert	Novice (N)	Novice (Ne)	Novice
n1	37	42	41	
n2	35	45	46	
N1	177	234	173	
N2	285	289	297	
V	2020	2724	2482	
D	153	135	136	
E	478965	141034	146467	
V(G)	10	10	12	
H&K	29			25
TSO	3			2
DH	11			35
DA	2			5
N	461	461	469	
Ne	274	274	274	
L	.17	.54	.46	

(N) the values are extrapolated with program length.
(Ne) the values are extrapolated with estimated program length.

TABLE XXII

MEANS AMONG THE SELECTED METRICS FOR
NOVICES' AND EXPERTS' COMPARISON
FOR C++ PROGRAMS

	Expert	Novice (N)	Novice (Ne)	Novice
n1	45	98	49	
n2	31	71	35	
N1	244	245	129	
N2	404	404	209	
V	2910	2659	1377	
D	320	294	149	
E	1386527	312338	159509	
V(G)	15	19	9	
H&K	17			25
TSO	1			1
DH	10			23
DA	3			3
N	649	649	338	
Ne	286	579	286	
L	.11	.45	.21	

(N) the values are extrapolated with program length.

(Ne) the values are extrapolated with estimated program length.

Table XXII also shows the average of Henry and Kafura's and Cox's surface area metrics. It shows that, on the average, novices' programs passed and received data in the program modules more than the experts' counterpart. Cox's TSO metrics of the experts' and the novices' programs are the same. This is primarily because of the way an object is defined in C++. When the object is initiated, it is ready to be used and it is automatically destroyed when the object

is not in use. Table XXIII indicates that ,on the average, Cox's data abstraction of the experts' programs are also equal to the novices' programs. The only difference regarding Cox's surface area metrics between the experts' and the novices' programs is the data hiding. The novice programmers, on the average, used more global or external data than the experts. As mentioned before, data can be variables, constants, or data types (user or defined).

Novices' and Experts' Comparison Overview

The novices' programs were compared to the experts' programs by means of how close the novices' programs are to the experts'. Problems arose when the number of subjects and programs between the expert and novice groups differed. Extraction and duplication of the experts' programs were applied to obtain a match-up of the metrics. Table XXIII illustrated the adjustments.

After applying the correlation between the novices' Ada metrics and the corresponding experts' metrics, it was found that there was a linear relationship between the two sets of metrics except for Halsteads' number of operators and the volume metrics. The metrics applied were Halstead's and McCabe's metrics. In the C++ comparison between the experts and the novices, the correlation coefficients produced were better than the Ada counterpart. A good correlation between the experts' and the novices' metrics was unexpected since the novices' metrics were produced from extrapolated partial

programs. The extrapolation methods applied to the novices' metrics might be one of the reasons for this unexpected result.

TABLE XXIII

ILLUSTRATION OF EXPERTS' METRICS ADJUSTMENT

Program	Expert Metric	Novice Metric
PRG1	2	1
	2 <DUPLICATED>	3
	2 <DUPLICATED>	2
PRG2	4	3
	4 <DUPLICATED>	5
PRG3	5	6
	5 <DUPLICATED>	4
	5 <DUPLICATED>	7
PRG4	7 <EXTRACTED>	not available
PRG5	8 <EXTRACTED>	not available
PRG6	7	9
	7 <DUPLICATED>	8
	7 <DUPLICATED>	9

When the means of the Ada program level metric between the novices' and the experts' programs were compared, the metrics varied greatly. The metrics' ordinal values only

implied that the experts' programs were more complex and difficult to write than the novices' programs. The result was also true for the C++ program level analysis. In the Ada analysis, it was found that the experts' program modules passed and received data almost as much as the novices' program modules. However, in the C++ analysis, the novices' program modules passed and received more data than the experts' program modules. In the Ada programs, it was found that the expert programmers used more operations in using an object than the novice programmers. While in C++ programs, both the expert and the novice programmers needed only one operation to use an object. Both in Ada and C++ programs, the novice programmers defined more external data than the expert programmers.

Comparing the experts' programs and the novices' programs is a stimulating issue. Many questions can be posed regarding this evaluation. A general question would be "Did the experts produce better programs than the novices?". There are certainly many definitions of "better programs". But, in this study, "better programs" is defined as "less complex programs". The complexity of a program is measured by Halstead's, McCabe's, Henry and Kafura's, and Cox's metrics.

The discussion above tried to answer this question with the following factors in mind. First, the experts' and the novices' programs were obtained in different ways. Second, the novices' metrics were extrapolated values. Third, the

average programs were about 300 tokens. Finally, the programs were designed by using the object-oriented style. Even though the result is constrained to a small population with the above factors in mind, the result was intended to reflect the population with the first and the second factors removed.

Discussion

This chapter analyzed and evaluated the experts' and the novices' programs. A total of twenty-four object-oriented programs were obtained from various textbooks [2, 7, 16, 21, 33, 43, 52-56]. The programs were written in Ada, Modula-2, and C++ (eight programs for each language were considered). These programs were called the experts' programs. The novices' programs were obtained in a controlled experiment. Because of time and resource (no Modula-2 compiler was available) constraints, only four programs of the eight possible programs were implemented in Ada and C++ language for the experiment. Volunteers for the experiment were eleven students consisting of computer science graduate and advanced undergraduate students. From those eleven students, three used C language more than three semesters, four used it for two semesters, and four used it in one semester or less. When asked how long they used Pascal, six used it three semesters or more, one used it two semesters, and three used it one semester or less.

The programs produced by the expert and novice

programmers were measured by complexity analyzers. These analyzers (written in Ada and C++) read a program and produced Halsteads' and McCabe's metrics. Henry and Kafura's data sharing and Cox's surface area metrics were produced manually.

A correlation coefficient has been used to analyze and compare programs in similar studies. It was felt that this method of evaluation was good for this experiment. The Pearson correlation coefficients discussed in this chapter were obtained from feeding the metrics to SAS (Statistic Analysis System). SAS also produced the means of the metrics for each language. These average values were quite useful for comparison where a correlation coefficient between two metrics was not meaningful. In case of Halstead's program level, Cox's metrics, or metrics between two languages, the average value was proven to be better for comparison.

In this experiment, it was found that more than half of the targeted novice programs were not completed. To overcome this problem, the percentage and extrapolation methods were applied to the metrics. Even though the experts' and the novices' programs were not obtained in a similar way, the correlation coefficients and the average values of two metrics were still valuable to be evaluated. There was a problem in making this evaluation possible. The number of programs between the two groups of programmers were different because only four of the eight programs were implemented by the novices and also five to six novices

wrote similar problems. Extraction and duplication techniques were used to balance out the programs being compared.

There were three evaluations discussed in this chapter. They were the experts' program evaluation, the novices' program evaluation, and the experts' versus novices' program evaluation. At the end of every evaluation, the limitation factors were discussed. It was intended to eliminate misunderstandings pertaining to the interpretation of the results. Even with these limitation factors, it was felt that this study accomplished the intention of understanding the effect of the object-oriented style on developing programs.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

This thesis evaluates object-oriented programs written by expert and novice programmers. The comparison between the experts' and the novices' programs are also analyzed. The experts' programs were written in Ada, C++, and Modula-2. Eight programs for each language were collected from various books. These twenty-four programs are called the experts' programs. The novices' programs were obtained from a controlled experiment. These programs were written by eleven Oklahoma State University advanced undergraduate and graduate students. The students wrote four of the eight programs written by the experts. Ada, C++, and Modula-2 source code analyzers were developed to produce eleven selected metrics (Halstead's, McCabe's cyclomatic, Henry and Kafura's data sharing, and Cox's surface area metrics). Extrapolation and percentage techniques were applied to the novices' partial programs to normalize the partial metrics. Extraction and duplication methods were applied to the experts' metrics to compare the experts' and the novices' metrics.

In the expert evaluation, it was found that Halstead's metrics correlated well with McCabe's metric. That was true for all programs which were written in Ada, C++, and Modula-2. When the data sharing metric was observed, the expert Ada and Modula-2 programs were linearly related. It was also found that, on the average, C++ programs have the least shared data followed by Ada and then Modula-2 programs. As expected, Cox's TSO metric of C++ programs had the least sequence of operations while Ada and Modula-2 required almost the same number of operations. On the average, the expert Ada, C++, and Modula-2 data sharing metrics had the same ordinal number. As with the data sharing metric, the expert programmers used the same number of data abstraction in Ada, C++, and modula-2 programs.

There were two methods applied to the novices' programs: extrapolation and percentage methods. The extrapolation method attempted to estimate the metrics from the novices' partial programs. These metrics were then correlated as in the expert evaluation. The percentage method measured the percentage of the partial program over the expected finished program. When the extrapolated novice metrics and the percentage number of the novice programs were examined, it was clear that the novice programmers wrote better in Ada than in C++. From the number of unfinished programs, it was felt that the novice programmers needed more treatment in learning object-oriented programming.

When the percentage numbers were observed, no programmer produced a complete C++ program. Complete means a program without a compilation or run-time error. Three novice programmers finished over seventy-five percent and the rest finished below fifty percent. It was also found that six novice programmers finished at least one Ada program; three finished two programs; and five finished below fifty percent.

In this study, the partial metrics were extrapolated with Halstead's length metric. For comparison, they were also extrapolated with Halstead's estimated length metric. It was found that they produced almost similar correlation coefficients. The extrapolated Halstead's metrics correlated well with their own member metrics. Unlike the expert Halstead's metrics, they did not correlate well with McCabe's metric. On the average, Henry and Kafura's metric of novice C++ programs was larger than Ada programs. As in the expert programs, the TSO metric of C++ had a lower number than the same metric of Ada. The novice programmers, on the average, used less data abstraction and less external data in C++ than in Ada.

Because of different numbers of expert and novice programmers as well as a different number of tasks written by both groups of programmers, extraction and duplication methods were applied to the expert metrics to balance out the metrics being compared. With the exception of the volume and the number of operations metrics, Halstead's and

McCabe's metrics of expert programs correlated well with the novice programs. It was found that the experts' and novices' Ada program modules, on the average, passed and received almost equal numbers of data. While in C++, the novices' program modules passed and received more data than the experts' program modules. In Ada programs, the novices used more data abstraction and more unprotected data than the experts. It was also found that the Ada expert programmers used more TSO than the novice programmers. This was partly caused by a good programming practice followed by the experts which was always destroying an object when it was not needed. In C++ programs, the expert and novice programmers used the same number of TSO and data abstraction. The novices, however, used more unprotected data than the experts in C++ programs.

When Halstead's language level was observed, both the experts' and the novices' metrics show that C++ has a lower language level than Ada. It needed to be pointed out that the language levels were not a fixed number. In fact, it can be proved that the language level is a decreasing function of the size metrics. It was also found that the Modula-2 language level was lower than Ada but higher than C++. Halstead hypothesized that lower language level meant more complex language. In general, both in the expert and the novice analysis, C++ programs were larger in size but smaller in surface area than Ada and Modula-2 programs. In the expert analysis, Modula-2 programs were larger in size

than Ada and C++ programs. In surface area, both Modula-2 and Ada metrics had almost the same number.

There are some factors that need to be discussed in order to avoid misunderstanding the results of this study. First, all of the programs were written in the object-oriented style. So, the result does not generalize to other styles of programming. Second, the experts' programs were selected from various textbooks while the novices' programs were obtained from a controlled experiment. Third, the average program size of the expert programs was about 750 tokens and 500 tokens for the novice programs. Fourth, the novice metrics were extrapolated metrics. Finally, eight expert programs for each language were evaluated and a total of sixteen novice programs were observed. So, this study might not reflect populations with larger program sizes and larger numbers of observations. Even with these limiting factors, it was felt that this study achieved its purpose of understanding the effect of the object-oriented style on developing programs. The results were felt to be worth analyzing and comparing to other similar studies.

Recommendations

There are several null hypotheses that could be formulated from this exploratory study. First, novices write programs faster in Ada than in C++. Second, C++ programs have lower surface area than Ada or Modula-2 programs. Third, novice C++ programs are larger than novice Ada

programs. Fourth, experts' programs have less surface area and size than novices' programs. Finally, expert Ada programs and Modula-2 programs are comparable in size and surface area.

Future studies could investigate several areas. A controlled study with a larger population of subjects would be helpful. Other object-oriented languages, such as Smalltalk, Flavors, and Loops could be used in addition to Ada, C++, and Modula-2. The hyphotheses formed as a consequence of this initial pilot study could be tested. Problems that exploit inheritance could be included. Finally, Syntactic and semantic differences among the object-oriented programming languages pertaining to issues (including understandability and relative complexity), that may justify or explain the differences demonstrated by the controlled experiment could be investigated.

A SELECTED BIBLIOGRAPHY

1. Atkinson, M.P. and Morrison, R., Procedures as Persistent Data Objects, ACM Transaction on Programming Languages and Systems 7, 4 (October 1985), 539-559.

2. Barnes, J.G.P., Programming in Ada, Addison-Wesley, 1982.

3. Basili, V.R., Selby, R.W., and Hutchens, D.H., Experimentation in Software Engineering, IEEE Transaction on Software Engineering SE-12, 7 (July 1986), 733-741.

4. Beidler, J. and Jackowitz, P., Consistent Generics in Modula-2, ACM Sigplan Notices 21, 4 (April 1986), 32-41.

5. Booch, G., Object Oriented Development, IEEE Transactions on Software Engineering SE-12, 2 (February 1986), 211-220.

6. Booch, G., Reusability : The Case for Object-Oriented Design, IEEE Software (March 1987), 50-221.

7. Booch, G., Software Engineering with Ada, Benjamin / Cummings, Menlo Park, CA, 1983.

8. Book, S.A., Statistics : Basic Techniques for Solving Applied Problems, McGraw-Hill, Inc., New York, 1977.

9. Brooks, R.E., Studying Programmer Behavior Experimentally: The Problems of Proper Methodology, Communications of the ACM 23, 4 (1980), 207-213.

10. Buzzard, G.D. and Mudge, T.N., Object-Based Computing and the Ada Programming Language, IEEE Computer (March 1985), 11-18.

11. Cargill, T.A., Pi : A Case Study in Object-Oriented Programming, OOPSLA'86 Proceedings (September 1986), 350-360.

12. Cmelik, R.F. and Gehani, N.H., Dimensional Analysis with C++, IEEE Software (May 1988), 21-26.

13. Coar, D., Pascal, Ada, and Modula-2, BYTE (August 1984), 361-367.

14. Conte, S.D., Dunsmore, H.E., and Shen, V.Y., Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park, CA, 1986.

15. Cox, B.J., Message/Object Programming : An Evolutionary Change in Programming Technology, IEEE Software (January 1984), 50-61.

16. Cox, B.J., Object-Oriented Programming - An Evolutionary Approach, Productivity Products International, Sansy Hook, CT, 1983.

17. Crawford, S.G., McIntosh, A.A., and Pregibon, D., An Analysis of Static Metric and Fault in C Software, The Journal of Systems and Software 5 (1985), 37-48.

18. Cunningham, W. and Beck, K., A Diagram for Object-Oriented Programs, OOPSLA'86 Proceedings (September 1986), 361-367.

19. Dewhurst, S.C. and Stark, K.T., Out of the C World Comes C++, Computer Language (February 1987), 29-36.

20. Donahue, J. and Demers, A., Data Types are Values, ACM Transaction on Programming Languages and Systems 7, 3 (July 1985), 426-445.

21. Ford, G.A. and Wiener, R.S., Modula-2: An Approach to Software Development, John Wiley & Sons, 1986.

22. Ford, R. and Miller, K., Abstract Data Type Development and Implementation : an Example, IEEE Transactions on Software Engineering SE-11, 10 (October 1985), 1033-1037.

23. Gannon, J.D., Katz, E.E., and Basili, V.R., Metrics for Ada Packages: An Initial Study, Communication of the ACM 29, 7 (July 1986), 616-623.

24. Gargaro, A., Reusability Issues and Ada, IEEE Software (July 1987), 43-51.

25. Gehani, N.H., Ada's Derived Types and Units of Measure, Software-Practice and Experience 15, 6 (June 1985), 555-569.

26. Halbert, D.C. and O'Brien, P.D, Using Types and Inheritance in Object-Oriented Programming, IEEE Software (September 1987), 71-79.

27. Kafura, D. and Geereddy R.R., The Use of Software Complexity Metrics in Software Maintenance, IEEE Transaction on Software Engineering SE-13, 3 (March 1987), 335-343.

28. Katwijk, J.V., Addressing Types and Objects in Ada, Software-Practice and Experience 17, 5 (May 1987), 319-343.

29. Krogdahl, S. and Olsen, K.A., Ada, as Seen from Simula, Software-Practice and Experience 16, 8 (August 1986), 689-700.

30. Li, H.F. and Cheung, W.K., An Empirical Study of Software Metrics, IEEE Transaction on Software Engineering SE-13, 6 (Junne 1987), 697-708.

31. Meyer, B., Genericity Versus Inheritance, OOPSLA'86 Proceedings (September 1986), 391-405.

32. Mitchell, J., Urban, J.E., and McDonald, R., The Effect of Abstract Data Types on Program Development, IEEE Computer (August 1987), 85-88.

33. Mohnkern, G.L., Mohnkern, B., Applied Ada, Tab Books Inc., 1986.

34. Oktaba, H. and Berber, R., Crafting Reusable Software in Modula-2, BYTE (September 1987), 123-128.

35. Pomberger, G., Software Engineering and Modula-2, Petrocelli Books, Princeton, NJ, 1985.

36. Ramamurthy, B. and Melton, A., A Synthesis of Software Science Measures and the Cyclomatic Number, IEEE Transactions on Software Engineering 14, 8 (August 1988), 1116-1121.

37. Rascoe, G.A., Elements of Object-oriented Programming, BYTE (August 1986), 139-144.

38. Reynolds, R.G., The Partial Metrics System: Modeling the Stepwise Refinement Process Using Partial Metrics, Communication of the ACM 30, 11 (November 1987), 956-963.

39. Reynolds, R.G., Metrics to Measure the Complexity of Partial Programs, The Journal of Systems and Software 4 (1984), 75-91.

40. Sammet, J.E., Why Ada is not Just Another Programming Language, Communications of the ACM 29, 8 (August 1986), 722-732.

41. Shammas, N.C., Exploring Ada and Modula-2, Computer Language (1984), 51-60.

42. Snyder, A., Encapsulation and Inheritance in Object-Oriented Programming Languages, OOPSLA'86 Proceedings (September 1986), 38-45.

43. Stroustrup, B., The C++ Programming Language, AT&T Bell Laboratories, Murray Hill, NJ, 1986.

44. Stroustrup, B., What is Object-Oriented Programming?, IEEE Software (May 1988), 10-20.

45. Trickey, H., C++ Versus Lisp : A Case Study, ACM Sigplan Notices 22, 6 (June 1987), 59-68.

46. Tucker, A.B., Computer Science: A Second Course Using Modula-2, McGraw-Hill, Inc., New York, 1988.

47. Unger, B.W., Object-Oriented Simulation - Ada, C++, and Simula, Proceedings of the 1986 winter Simulation (1986), 123-124.

48. Vasilescu, E.N., Ada: Programming with Applications, Wm. C. Brown Publishers, Dubuque, Iowa, 1988.

49. Wegmann, R., Object-Oriented Programming Using Modula-2, Journal of Pascal, Ada, and Modula-2 (May/June 1986), 5-17.

50. Wiedenbeck, S., Novice/Expert Differences in Programming Skills, Int. J. Man-Machine Studies 23 (1985), 383-390.

51. Wiener, R.S., Object-Oriented Programming in C++ - A Case Study, Sigplan Notices 22, 6 (June 1987), 59-68.

52. Wiener, R. and Sincovec, R., Programming in Ada, John Wiley & Sons, New York, NY, 1983.

53. Wiener, R. and Sincovec, R., Software Engineering with Modula-2 and Ada, Wiley, New York, NY, 1984.

54. Wiener, R.S. and Pinson, L.J., An Introduction to Object-Oriented Programming and C++, Addison-Wesley, New York, NY, 1988.

55. Wirth, N., Programming in Modula-2, Springerverlag, Berlin, Heidelberg, 1985.

56. Young, S.J., An Introduction to Ada, John Wiley & Sons, New York, 1984.

APPENDIXES

APPENDIX A

PROBLEM SPECIFICATIONS

Task 1a - MATRIX and COMPLEX

Design object MATRIX that implements the following operations:

- add two matrixs
- subtract two matrixs
- multiply two matrixs.

Design object COMPLEX that implements the following operations:

- add two complex numbers
- subtract of two complex numbers
- multiply of two complex numbers
- divide of two complex numbers.

Definition of complex number

A Complex number is an ordered pair of numbers. The first number of the pair is commonly called the "real part" and the second one the "imaginary part". Complex numbers are commonly written as:

A + iB,

where A is the real part, B is the imaginary part and i is the square root of 1; for computational purposes,however, it is more common to write complex numbers as :

A,B

For a pair of complex numbers, C = (A,B) and D = (E,F), the complex operations are defined as:

Addition : C + D = (A,B) + (E,F) = (A+E,B+F)

subtraction: C - D = (A,B) - (E,F) = (A-E,B-F)

Multiplication: C * D = (A,B) * (E,F)

= (A*E - B*F, A*F + B*E) and

Division: C / D = (A,B) / (E,F)

= (A*E + B*F, B*E - A*F) /

(E**2 + F**2, 0)

= ((A*E + B*F)/(E**2 + F**2),

(B*E - A*F)/(E**2 + F**2))

Task 1b - LIST and QUEUE

Design object LIST to represent singly link list data structure that has the following operations:

- insert integer number (add to the front of the list)
- append integer number (add to the back of the list)
- remove integer number (remove the head of the list)
- clear (remove all the nodes in list).

Design object QUEUE to represent queue data structure that has the following operations:

- put integer number (add to the front of the list)
- get integer number (remove the head of the list).

Task 2 - TABLE

Design object TABLE that implements tree data structure. The object has the following operations:

- insert a string into the table
- display all the string inside the table
  (use traverse in-order).

APPENDIX B

MAIN DRIVERS FOR C++ PROGRAMS

```
 // matrix driver
#include "matrix.h"
#include <stream.h>

main()
{
   matrix a;
   matrix b;

   // assign value into matrix a and b
   a.m[1,1] = 1;    b.m[1,1] = 2;
   a.m[1,2] = 2;    b.m[1,2] = 3;
   a.m[1,3] = 3;    b.m[1,3] = 4;
   a.m[2,1] = 2;    b.m[2,1] = 1;
   a.m[2,2] = 3;    b.m[2,2] = 2;
   a.m[2,3] = 4;    b.m[2,3] = 3;
   a.m[3,1] = 3;    b.m[3,1] = 4;
   a.m[3,2] = 3;    b.m[3,2] = 4;
   a.m[3,3] = 1;    b.m[3,3] = 2;

   a.display;
   b.display;

   matrix c;
   c = a + b;
   c.display;

   matrix c;
   c = a * b;
   c.display;

}
```

```
// complex number driver
#include "complex.h"
#include <stream.h>

// suggested output of complex number
ostream& operator << (ostream &s, complex z)
{
     return s << "(" << real(z) << imag(z) << ")";
}

main()
{
   complex a(4,5);
   complex b(-10,6);
   complex c(0,0);

   cout << "\na = " << a;
   cout << "\nb = " << b;

   c = a + b;
   cout << "\nc = a + b = " << c;

   c = a - b;
   cout << "\nc = a - b = " << c;

   c = a * b;
   cout << "\nc = a * b = " << c;

   c = a / b;
   cout << "\nc = a / b = " << c;
}
```

```
// list, queue driver
#include <stream.h>
#include "list.h"
#include "listqe.h"

main()
{
   int a;

   int_list mylist(10);   // size of mylist is 10
   cout << "\n LIST "     // test object list
   a = 1;
   mylist.insert(a);
   a = 2;
   mylist.insert(a);
   a = 3;
   mylist.append(a);
   a = 4;
   mylist.append(a);

   a = mylist.get();
   cout << a;
   a = mylist.get();
   cout << a;
   a=mylist.get();
   cout << a;
   a=mylist.get();
   cout << a;

   int_queue myqueue;
   cout << "\n QUEUE "   // test object queue
   a = 1;
   myqueue.put(a);
   a = 2;
   myqueue.put(a);
   a = 3;
   myqueue.put(a);

   a = myqueue.get();
   cout << a;
   a = myqueue.get();
   cout << a;
   a = myqueue.get();
   cout << a;
}
```

```
// table driver
#include "table.h"

main()
{
    table table1;
    table table2;

    table1.insert("ada");
    table1.insert("mod");
    table1.insert("cplusplus");
    table1.insert("basic");
    table1.insert("modulatwo");

    table2.insert("plone");
    table2.insert("lisp");
    table2.insert("fortran");
    table2.insert("smalltalk");

    table1.display();
    table2. display();
}
```

APPENDIX C

MAIN DRIVERS FOR ADA PROGRAMS

```
-- matrix driver
with MATRIX, TEXT_IO;
use MATRIX;

procedure MATRIX_TEST is
   A : MATRIX;
   B : MATRIX;
   C : MATRIX;

   -- assign value into matrix a and b
   A(1,1) := 1;    B(1,1) = 2;
   A(1,2) := 2;    B(1,2) = 3;
   A(1,3) := 3;    B(1,3) = 4;
   A(2,1) := 2;    B(2,1) = 1;
   A(2,2) := 3;    B(2,2) = 2;
   A(2,3) := 4;    B(2,3) = 3;
   A(3,1) := 3;    B(3,1) = 4;
   A(3,2) := 3;    B(3,2) = 4;
   A(3,3) := 1;    B(3,3) = 2;

   TEXT_IO.PUT (" Matrix A is "); OUTPUT_MATRIX(A);
   TEXT_IO.PUT (" Matrix B is "); OUTPUT_MATRIX(B);

   C := A + B;
   TEXT_IO.PUT (" C = A + B = "); OUTPUT_MATRIX(C);

   C : = A * B;
   TEXT_IO.PUT (" C = A + B = "); OUTPUT_MATRIX(C);

end MATRIX_TEST;
```

```
-- complex number driver
with COMPLEX, TEXT_IO;
use COMPLEX;

procedure COMPLEX_TEST is
   A : COMPLEX_TYPE;
   B : COMPLEX_TYPE;
   C : COMPLEX_TYPE;
begin
   A.Re := 4;      -- real part of A
   A.Im := 5;      -- imiginary part of A
   B.Re := -10;
   B.Im := 6;

   TEXT_IO.PUT ("a = "); OUTPUT_COMPLEX(A);
   TEXT_IO.PUT ("b = "); OUTPUT_COMPLEX(B);

   C := A + B;
   TEXT_IO.PUT ("c = a + b = "); OUTPUT_COMPLEX(C);

   C := A - B;
   TEXT_IO.PUT ("c = a - b = "); OUTPUT_COMPLEX(C);

   C := A * B;
   TEXT_IO.PUT ("c = a * b = "); OUTPUT_COMPLEX(C);

   C := A / B;
   TEXT_IO.PUT ("c = a / b = "); OUTPUT_COMPLEX(C);

end COMPLEX_TEST;
```

```
-- list, queue driver
with TEXT_IO, LISTS, QUEUES;

procedure LIST_QUEUE is
   package INT_IO is new TEXT_IO.INTEGER_IO(INTEGER);
   A : INTEGER;
   THE_LIST : LISTS.LIST;
   THE_QUEUE : QUEUES.QUEUE;

   TEXT_IO.PUT ("*** LIST TEST ***");
   A := 1; LISTS.ADD_HEAD_TO_LIST (A, THE_LIST);
   A := 2; LISTS.ADD_HEAD_TO_LIST (A, THE_LIST);
   A := 3; LISTS.ADD_TAIL_TO_LIST (A, THE_LIST);
   A := 4; LISTS.ADD_TAIL_TO_LIST (A, THE_LIST);

   LISTS.REMOVE_HEAD(A, THE_LIST); INT_IO.PUT_LINE (A);
   LISTS.REMOVE_HEAD(A, THE_LIST); INT_IO.PUT_LINE (A);
   LISTS.REMOVE_HEAD(A, THE_LIST); INT_IO.PUT_LINE (A);
   LISTS.REMOVE_HEAD(A, THE_LIST); INT_IO.PUT_LINE (A);

   TEXT_IO.PUT ("*** QUEUE TEST ***");
   A := 1; QUEUES.JOIN (A, THE_QUEUE);
   A := 2; QUEUES.JOIN (A, THE_QUEUE);
   A := 3; QUEUES.JOIN (A, THE_QUEUE);
   A := 4; QUEUES.JOIN (A, THE_QUEUE);

   QUEUES.REMOVE (A, THE_QUEUE);   INT_IO.PUT_LINE (A);
   QUEUES.REMOVE (A, THE_QUEUE);   INT_IO.PUT_LINE (A);
   QUEUES.REMOVE (A, THE_QUEUE);   INT_IO.PUT_LINE (A);
   QUEUES.REMOVE (A, THE_QUEUE);   INT_IO.PUT_LINE (A);

end LIST_QUEUE;
```

```
-- table driver
with TABLES;

procedure TABLE_TEST is

    TABLES.INSERT("ada");
    TABLES.INSERT("mod");
    TABLES.INSERT("cplusplus");
    TABLES.INSERT("basic");
    TABLES.INSERT("modulatwo");
    TABLES.INSERT("plone");
    TABLES.INSERT("lisp");
    TABLES.INSERT("fortran");
    TABLES.INSERT("smalltalk");

    TABLES.DISPLAY();

end TABLE_TEST;
```

APPENDIX D

POST-TEST QUESTIONNAIRE

Post-experiment Questionnaire

Group: _____ ID #: _____

1) Knowledge level of Pascal: (circle one)
I have programmed in Pascal for

(a) one semester (b) two semesters (c) three semesters

(d) more than three semesters (e) every day

2) Knowledge level of C: (circle one)
I have programmed in C for

(a) one semester (b) two semesters (c) three semesters

(d) more than three semesters (e) every day

3) On a scale of 1 - 5 (lowest to highest) rate the understandability of

(a) type definitions in Ada _____

(b) type definitions in C++ _____

(c) functions and procedures in Ada _____

(d) functions and procedures in C++ _____

(e) packages in Ada _____

(f) classes in C++ _____

4) On a scale of 1 - 5 (lowest to highest) rate the ease of use of

(a) type definitions in Ada _____

(b) type definitions in C++ _____

(c) functions and procedures in Ada _____

(d) functions and procedures in C++ _____

(e) packages in Ada _____

(f) classes in C++ _____

5) On a scale of 1 - 5 (lowest to highest) how do you rate Ada and C++ in the following categories?

	Ada	C++
(a) personal preference as a programming language	_____	_____
usefulness of diagnostic messages		
(b) compilation	_____	_____
(c) run time	_____	_____

6) On a scale of 1 - 5 (lowest to highest) how do you rate the tasks in the following categories?

(a) appropriateness for application of object oriented techniques	_____	_____
(b) sufficiency to distinguish between Ada and C++	_____	_____
(c) difficulty of task #1	_____	_____
(d) difficulty of task #2	_____	_____
(e) availability of resources		
technical assistance	_____	_____
manuals and reference material	_____	_____

7) Comments and suggestions:

(a) tasks

(b) languages used

(c) experiment

(d) etc.

APPENDIX E

PROGRAM TO CALCULATE HALSTEAD'S AND McCABE'S METRICS OF ADA PROGRAMS

```
T''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'     This program reads an Ada program,                      '
'                  parses the program into operands and       '
'                  operators,                                 '
'                  inserts them into a symbol table, and      '
'                  calculates the software metrics and        '
'                  McCabe's metric from the symbol table.     '
'                                                             '
'     Author : Budy Tjahjo                                    '
'     updated: Dec 10, 1988                                   '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

DEFINT A-Z
TYPE symbol
        word AS STRING * 15
        wordType AS INTEGER
        count AS INTEGER
END TYPE
CONST EOFILE = -1
CONST OPERATOR = 1
CONST OPERAND = 2
DIM SHARED symTab(500) AS symbol                        ' symbol table
DIM SHARED nTable                                       ' number of symbols
DIM SHARED delim$(0 TO 31)                              ' delimeters
DIM SHARED key$(1 TO 78)                                ' keywords
DIM SHARED lastEnt                                      ' last record number
DIM SHARED commentFlag                                  ' comment toggle

'BEGIN
       CALL loadKey                                     ' load the Ada keywords
       DO                                               ' loop until no more file
           CLS
           INPUT "Enter the filename "; filename$       ' get the filename
           CALL strip(filename$)
           IF LEN(filename$) = 0 THEN EXIT DO           ' no more file

           CALL initialize(filename$)                   ' open the file

           CALL readString(rec$, mode)                  ' read record from file
           WHILE (mode <> EOFILE)                       ' loop until eof
                CALL strip(rec$)
                IF LEN(rec$) > 0 THEN CALL insert(rec$)    ' parse and insert
                CALL readString(rec$, mode)                ' read the next record
           WEND
           CALL clearAll
       LOOP

       LPRINT "******* H a l s t e a d ******** "
       CALL halstead
       LPRINT "******* McCabe complexity *********"
       CALL mcCabe
END
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    close the file when finish processing          '
''''''''''''''''''''''''''''''''''''''''''''''''''''''

SUB clearAll
       CLOSE #1
END SUB



''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   get the position of delimeter inside word$       '
''''''''''''''''''''''''''''''''''''''''''''''''''''''

SUB getDelimPos (word$, del$, post)
     post = 0
     DO
           IF index > 31 THEN post = 0: EXIT SUB
           del$ = delim$(index)
           post = INSTR(word$, del$)
           index = index + 1
     LOOP UNTIL post > 0
END SUB



''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' calculate the halstead's metrics                         '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

SUB halstead
      FOR i = 1 TO nTable
          IF symTab(i).wordType = OPERAND THEN
                  smallN1 = smallN1 + 1
                  bigN1 = bigN1 + symTab(i).count
          ELSE
                   smallN2 = smallN2 + 1
                  bigN2 = bigN2 + symTab(i).count
          END IF
      NEXT
      LPRINT "number of unique operators : "; smallN1
      LPRINT "number of unique operands : "; smallN2
      LPRINT "total occurences of operators : "; bigN1
      LPRINT "total occurences of operands : "; bigN2
      LPRINT "length : "; bigN1 + bigN2
      LPRINT "Estimate Length : "; (smallN1 * LOG(smallN1)) + (smallN2 * LOG(smallN2))
      smallN = smallN1 + smallN2
      bigN = bigN1 + bigN2
      V# = bigN * LOG(smallN)
      LPRINT "volume : "; V#
      impLevel# = (2 * smallN2) / (smallN1 * bigN2)
      impEffort# = V# / impLevel#
      langLevel# = (impLevel# * impLevel#) * V#
      LPRINT "Level of Implementation of a program : "; impLevel#
      LPRINT "The difficulty :"; 1 / impLevel#
```

```
      LPRINT "Implementation Effort : "; impEffort#
      LPRINT "Level of programming language : "; langLevel#
      LPRINT : LPRINT
END SUB


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  open the file to start                                                       '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB initialize (filename$)
     OPEN "i", #1, filename$
END SUB


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    parse and insert the keywords inside the rec$ into                         '
'    symbol table                                                               '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB insert (rec$)
        DIM word$(50)

        CALL split(rec$, word$(), nword)
        FOR i = 1 TO nword
             IF word$(i) = "--" THEN EXIT SUB
             wordType = isKeyWord(word$(i))
             IF wordType = OPERATOR THEN
                    CALL insert2SymbTab(word$(i), OPERATOR)
             ELSE
                    CALL insert2SymbTab(word$(i), OPERAND)
             END IF
        NEXT
        EXIT SUB
END SUB


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   insert the string w$ and its type into symbo table                          '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB insert2SymbTab (w$, wordType)
        IF commentFlag THEN EXIT SUB
        index = lookUp(w$)
        IF index > nTable THEN
             nTable = index
             symTab(index).word = UCASE$(w$)
             symTab(index).wordType = wordType
        END IF
        symTab(index).count = symTab(index).count + 1
END SUB
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   return the type of the word$ (OPERATOR or OPERAND)   '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
FUNCTION isKeyWord (word$)
     FOR index = 1 TO 78
          IF UCASE$(word$) = key$(index) THEN
                 isKeyWord = OPERATOR
                 EXIT FUNCTION
          END IF
     NEXT
     FOR index = 0 TO 31
          IF word$ = delim$(index) THEN
                 isKeyWord = OPERATOR
                 EXIT FUNCTION
          END IF
     NEXT
     isKeyWord = OPERAND
END FUNCTION


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'     load the keywords and delimeters                   '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB loadKey
           key$(1) = "ABORT"
           key$(2) = "ABS"
           key$(3) = "ACCEPT"
           key$(4) = "ACCESS"
           key$(5) = "ADDRESS"
           key$(6) = "ALL"
           key$(7) = "AND"
           key$(8) = "ARRAY"
           key$(9) = "AT"
           key$(10) = "BASE"
           key$(11) = "BEGIN"
           key$(12) = "BODY"
           key$(13) = "CASE"
           key$(14) = "CONSTANT"
           key$(15) = "CONSTRAINED"
           key$(16) = "COUNT"
           key$(17) = "DECLARE"
           key$(18) = "DELAY"
           key$(19) = "DELTA"
           key$(20) = "DIGITS"
           key$(21) = "DO"
           key$(22) = "ELSE"
           key$(23) = "ELSIF"
           key$(24) = "END"
           key$(25) = "ENTRY"
           key$(26) = "EXCEPTION"
```

```
key$(27) = "EXIT"
key$(28) = "FIRST"
key$(29) = "FOR"
key$(30) = "FUNCTION"
key$(31) = "GENERIC"
key$(32) = "GOTO"
key$(33) = "IF"
key$(34) = "IMAGE"
key$(35) = "IN"
key$(36) = "IS"
key$(37) = "LAST"
key$(38) = "LENGTH"
key$(39) = "LIMITED"
key$(40) = "LOOP"
key$(41) = "MOD"
key$(42) = "NEW"
key$(43) = "NOT"
key$(44) = "NULL"
key$(45) = "OFS"
key$(46) = "OR"
key$(47) = "OTHERS"
key$(48) = "OUT"
key$(49) = "PACKAGE"
key$(50) = "POS"
key$(51) = "POSITION"
key$(52) = "PRAGMA"
key$(53) = "PRED"
key$(54) = "PRIVATE"
key$(55) = "PROCEDURE"
key$(56) = "RAISED"
key$(57) = "RANGE"
key$(58) = "RECORD"
key$(59) = "REM"
key$(60) = "RENAMES"
key$(61) = "RETURN"
key$(62) = "REVERSE"
key$(63) = "SELECT"
key$(64) = "SEPARATE"
key$(65) = "SIZE"
key$(66) = "SUBTYPE"
key$(67) = "SUCC"
key$(68) = "TASK"
key$(69) = "TERMINATE"
key$(70) = "THEN"
key$(71) = "TYPE"
key$(72) = "USE"
key$(73) = "VAL"
key$(74) = "VALUE"
key$(75) = "WHEN"
key$(76) = "WHILE"
key$(77) = "WITH"
key$(78) = "XOR"
```

```
        delim$(0) = "--"
        delim$(1) = "/="
        delim$(2) = "**"
        delim$(3) = "<>"
        delim$(4) = "<="
        delim$(5) = ">="
        delim$(6) = "=>"
        delim$(7) = ":="
        delim$(8) = ".."
        delim$(9) = "+"
        delim$(10) = "-"
        delim$(11) = "*"
        delim$(12) = "/"
        delim$(13) = "&"
        delim$(14) = "'"
        delim$(15) = "."
        delim$(16) = ","
        delim$(17) = "("
        delim$(18) = ")"
        delim$(19) = "["
        delim$(20) = "]"
        delim$(21) = "{"
        delim$(22) = "}"
        delim$(23) = "^"
        delim$(24) = "~"
        delim$(25) = "="
        delim$(26) = "#"
        delim$(27) = "<"
        delim$(28) = ">"
        delim$(29) = ":"
        delim$(30) = "|"
        delim$(31) = ";"
END SUB

''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  return the index of the word$ inside the symbol table'
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
FUNCTION lookUp (word$)
    index = 1
    WHILE index <= nTable
          temp$ = symTab(index).word
          CALL strip(temp$)
          IF temp$ = UCASE$(word$) THEN
               lookUp = index
               EXIT FUNCTION
          ELSE
               index = index + 1
          END IF
    WEND
    lookUp = index
END FUNCTION
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  calculate the McCabe's metric and print it          '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB mcCabe
      count = 0
      FOR i = 1 TO nTable
          key$ = symTab(i).word
          CALL strip(key$)
          IF key$ = "|" THEN count = count + symTab(i).count
          IF key$ = "&" THEN count = count + symTab(i).count
          IF key$ = "AND" THEN count = count + symTab(i).count
          IF key$ = "OR" THEN count = count + symTab(i).count
          IF key$ = "PROCEDURE" THEN count = count + symTab(i).count
          IF key$ = "CASE" THEN count = count + symTab(i).count
          IF key$ = "FOR" THEN count = count + symTab(i).count
          IF key$ = "MODULE" THEN count = count + symTab(i).count
          IF key$ = "REPEAT" THEN count = count + symTab(i).count
          IF key$ = "WHILE" THEN count = count + symTab(i).count
          IF key$ = "IMPLEMENTATION" THEN count = count + symTab(i).count
          IF key$ = "DEFINITION" THEN count = count + symTab(i).count
      NEXT
      LPRINT "v(G) = "; count + 1
END SUB



''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' read record from the file                            '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB readString (line$, mode)

      IF EOF(1) THEN
           mode = EOFILE
      ELSE
           INPUT #1, line$
           PRINT line$
           mode = 0
      END IF

END SUB
```

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' split the rec$ into keywords and insert them into     '
' word$ array. nword is the number of keywords found    '
' inside the rec$                                       '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB split (rec$, word$(), nword)
      DIM wordLine$(15)

      DO    ' split into word seperated by a blank
           CALL strip(rec$)
           IF LEN(rec$) = 0 THEN EXIT DO
           post = INSTR(rec$, " ")
           nwl = nwl + 1
           IF post = 0 THEN
                wordLine$(nwl) = rec$
           ELSE
                wordLine$(nwl) = LEFT$(rec$, post - 1)
                rec$ = MID$(rec$, post + 1, LEN(rec$))
           END IF
      LOOP UNTIL post = 0

      nword = 0
      FOR i = 1 TO nwl             ' spilt into words seperated by delim
           rec$ = wordLine$(i)
           DO
                CALL getDelimPos(rec$, del$, post)
                IF post = 0 THEN
                      nword = nword + 1
                      word$(nword) = rec$
                ELSE
                      IF post > 1 THEN
                          nword = nword + 1
                          word$(nword) = MID$(rec$, 1, post - 1)
                      END IF
                      nword = nword + 1
                      word$(nword) = del$
                      IF del$ = "--" THEN EXIT SUB
                      IF post = LEN(rec$) THEN EXIT DO
                      rec$ = MID$(rec$, post + 1, LEN(rec$))
                END IF
           LOOP UNTIL post = 0
      NEXT
END SUB


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' strip the blank of the word$                          '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB strip (word$)
     word$ = RTRIM$(word$)
     word$ = LTRIM$(word$)
END SUB
```

APPENDIX F

PROGRAM TO CALCULATE HALSTEAD'S AND McCABE'S METRICS OF C++ PROGRAMS

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  This program reads a C++ program, parses it into operators '
'     and operands, from the operatorsand operands, produce   '
'     the Halsteads' and McCabe's metrics                      '
'                                                              '
'  Author: Budy Tjahjo                                         '
'  Update: Dec 10, 1988                                        '
'                                                              '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'BEGIN
        DEFINT A-Z

        TYPE symbol
             word AS STRING * 15
             wordType AS INTEGER
             count AS INTEGER
        END TYPE
        CONST EOFILE = -1
        CONST OPERATOR = 1
        CONST OPERAND = 2

        DIM SHARED symTab(500) AS symbol          ' symbol table
        DIM SHARED nTable                         ' number of keywords
        DIM SHARED delim$(41)                     ' delimeters
        DIM SHARED key$(41)                       ' keywords
        DIM SHARED lastEnt                        ' last record number
        DIM SHARED commentFlag                    ' comment toggle

        CALL loadKey                              ' load keywords
        DO
            CLS
            INPUT "Enter the filename "; filename$        ' get the file
            CALL strip(filename$)
            IF LEN(filename$) = 0 THEN EXIT DO            ' no more file

            CALL initialize(filename$)                    ' get the file ready

            CALL readString(rec$, mode)                   ' read record form the file
            WHILE (mode <> EOFILE)
                  CALL strip(rec$)
                  IF LEN(rec$) > 0 THEN CALL insert(rec$)  ' parse and insert
                  CALL readString(rec$, mode)             ' read the next record
            WEND
            CALL clearAll
        LOOP
        LPRINT "******* H a l s t e a d ******** "
        CALL halstead

        LPRINT "******* McCabe complexity *********"
        CALL mcCabe
END
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   close the file when finish processing            '
''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB clearAll
      CLOSE #1
END SUB


'''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  get the position of delimeter inside word$         '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB getDelimPos (word$, del$, post)
     post = 0
     DO
          index = index + 1
          IF index > 41 THEN post = 0: EXIT SUB
          post = INSTR(word$, delim$(index))
     LOOP UNTIL post > 0
     del$ = delim$(index)
END SUB


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' calculate the halstead's metrics                         '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB halstead
      FOR i = 1 TO nTable
          IF symTab(i).wordType = OPERAND THEN
                  smallN1 = smallN1 + 1
                  bigN1 = bigN1 + symTab(i).count
          ELSE
                  smallN2 = smallN2 + 1
                  bigN2 = bigN2 + symTab(i).count
          END IF
      NEXT
      LPRINT "number of unique operators : "; smallN1
      LPRINT "number of unique operands : "; smallN2
      LPRINT "total occurences of operators : "; bigN1
      LPRINT "total occurences of operands : "; bigN2
      smallN = smallN1 + smallN2
      bigN = bigN1 + bigN2
      V# = bigN * LOG(smallN)
      LPRINT "v : "; V#
      impLevel# = (2 * smallN2) / (smallN1 * bigN2)
      impEffort# = V# / impLevel#
      langLevel# = (impLevel# * impLevel#) * V#
      LPRINT "Level of Implementation of a program : "; impLevel#
      LPRINT "The difficulty :"; 1 / impLevel#
      LPRINT "Implementation Effort : "; impEffort#
      LPRINT "Level of programming language : "; langLevel#
      LPRINT : LPRINT
END SUB
```

```
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  open the file to start                                                       '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB initialize (filename$)
    OPEN "i", #1, filename$
END SUB



'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    split the rec$ into keywords and insert each keyword                       '
'       into symbol table                                                       '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB insert (rec$)
        DIM word$(50)

        CALL split(rec$, word$(), nword)
        FOR i = 1 TO nword
             wordType = isKeyWord(word$(i))
             IF wordType = OPERATOR THEN
                    IF word$(i) = "/*" THEN
                         commentFlag = -1
                    ELSEIF word$(i) = "*/" THEN
                         commentFlag = 0
                    ELSE
                         CALL insert2SymbTab(word$(i), OPERATOR)
                    END IF
             ELSE
                    CALL insert2SymbTab(word$(i), OPERAND)
             END IF
        NEXT
        EXIT SUB
END SUB



'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'       insert w$ and the type into symbol table                                '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB insert2SymbTab (w$, wordType)
        IF commentFlag THEN EXIT SUB
        index = lookUp(w$)
        IF index > nTable THEN
             nTable = index
             symTab(index).word = UCASE$(w$)
             symTab(index).wordType = wordType
        END IF
        symTab(index).count = symTab(index).count + 1
END SUB
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'     return OPERATOR if the word$ is operator or              '
'            OPERANDS otherwise                                 '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
FUNCTION isKeyWord (word$)
      FOR index = 1 TO 41
           IF UCASE$(word$) = key$(index) THEN
                  isKeyWord = OPERATOR
                  EXIT FUNCTION
           END IF
      NEXT
      FOR index = 1 TO 41
           IF word$ = delim$(index) THEN
                  isKeyWord = OPERATOR
                  EXIT FUNCTION
           END IF
      NEXT
      isKeyWord = OPERAND
END FUNCTION


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'     load the keywords into key$() array and opreators '
'     and delimeters into delim$() array                 '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB loadKey
            key$(1) = "ASM"
            key$(2) = "AUTO"
            key$(3) = "BREAK"
            key$(4) = "CASE"
            key$(5) = "CHAR"
            key$(6) = "CLASS"
            key$(7) = "CONST"
            key$(8) = "CONTINUE"
            key$(9) = "DEFAULT"
            key$(10) = "DELETE"
            key$(11) = "DO"
            key$(12) = "DOUBLE"
            key$(13) = "ELSE"
            key$(14) = "ENUM"
            key$(15) = "EXTERN"
            key$(16) = "FLOAT"
            key$(17) = "FOR"
            key$(18) = "FRIEND"
            key$(19) = "GOTO"
            key$(20) = "IF"
            key$(21) = "INLINE"
            key$(22) = "INT"
            key$(23) = "LONG"
```

```
key$(24) = "NEW"
key$(25) = "OPERATOR"
key$(26) = "OVERLOAD"
key$(27) = "PUBLIC"
key$(28) = "REGISTER"
key$(29) = "RETURN"
key$(30) = "SHORT"
key$(31) = "SIZEOF"
key$(32) = "STATIC"
key$(33) = "STRUCT"
key$(34) = "SWICTH"
key$(35) = "THIS"
key$(36) = "TYPEDEF"
key$(37) = "UNION"
key$(38) = "UNSIGNED"
key$(39) = "VIRTUAL"
key$(40) = "VOID"
key$(41) = "WHILE"

delim$(1) = "//"
delim$(2) = "/*"
delim$(3) = "*/"
delim$(4) = "::"
delim$(5) = "->"
delim$(6) = "--"
delim$(7) = "++"
delim$(8) = "<<"
delim$(9) = "&&"
delim$(10) = ">>"
delim$(11) = "<="
delim$(12) = "!="
delim$(13) = "=="
delim$(14) = "||"
delim$(15) = ">="
delim$(16) = "~"
delim$(17) = "!"
delim$(18) = "+"
delim$(19) = "-"
delim$(20) = "*"
delim$(21) = "/"
delim$(22) = "&"
delim$(23) = "."
delim$(24) = ","
delim$(25) = "("
delim$(26) = ")"
delim$(27) = "["
delim$(28) = "]"
delim$(29) = "{"
delim$(30) = "}"
delim$(31) = "^"
delim$(32) = "%"
delim$(33) = "="
```

```
            delim$(34) = "#"
            delim$(35) = "<"
            delim$(36) = ">"
            delim$(37) = ":"
            delim$(38) = "?"
            delim$(39) = "|"
            delim$(40) = "@"
            delim$(41) = ";"
END SUB

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  return the table index of the keyword word$                            '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
FUNCTION lookUp (word$)
      index = 1
      WHILE index <= nTable
              temp$ = symTab(index).word
              CALL strip(temp$)
              IF temp$ = UCASE$(word$) THEN
                     lookUp = index
                     EXIT FUNCTION
              ELSE
                     index = index + 1
              END IF
      WEND
      lookUp = index
END FUNCTION



'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Calculate the McCabe's metric                                           '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB mcCabe
      FOR i = 1 TO nTable
          key$ = symTab(i).word
          CALL strip(key$)
          IF key$ = "IF" THEN count = count + symTab(i).count
          IF key$ = "WHILE" THEN count = count + symTab(i).count
          IF key$ = "FOR" THEN count = count + symTab(i).count
          IF key$ = "&&" THEN count = count + symTab(i).count
          IF key$ = "||" THEN count = count + symTab(i).count
      NEXT
      LPRINT "v(G) = "; count + 1

END SUB
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    get the record                                  '
''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB readString (line$, mode)

      IF EOF(1) THEN
           mode = EOFILE
      ELSE
           INPUT #1, line$
           mode = 0
      END IF
PRINT line$
END SUB

'''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   split the rec$ into keywords, insert them into    '
'     word$() array. nword is the number of keywords '
'     inside the word$()                              '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB split (rec$, word$(), nword)
       DIM wordLine$(30)

       DO    ' split into word seperated by a blank
            CALL strip(rec$)
            IF LEN(rec$) = 0 THEN EXIT DO
            post = INSTR(rec$, " ")
            nwl = nwl + 1
            IF post = 0 THEN
                 wordLine$(nwl) = rec$
            ELSE
                 wordLine$(nwl) = LEFT$(rec$, post - 1)
                 rec$ = MID$(rec$, post + 1, LEN(rec$))
            END IF
       LOOP UNTIL post = 0

       nword = 0
       FOR i = 1 TO nwl            ' spilt into words seperated by delim
            rec$ = wordLine$(i)
            DO
                 CALL getDelimPos(rec$, del$, post)
                 IF del$ = "//" THEN EXIT SUB
                 IF post = 0 THEN
                      nword = nword + 1
                      word$(nword) = rec$
                 ELSE
                      IF post > 1 THEN
                          nword = nword + 1
                          word$(nword) = MID$(rec$, 1, post - 1)
                      END IF
                      nword = nword + 1
                      word$(nword) = del$
```

```
                    IF post = LEN(rec$) THEN EXIT DO
                    rec$ = MID$(rec$, post + 1, LEN(rec$))
                END IF
            LOOP UNTIL post = 0
        NEXT
END SUB


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'     strip the blanks of the word$                                             '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB strip (word$)
     word$ = RTRIM$(word$)
     word$ = LTRIM$(word$)
END SUB
```

APPENDIX G

PROGRAM TO CALCULATE HALSTEAD'S AND McCABE'S METRICS OF MODULA-2 PROGRAMS

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'                                                        '
' This program reads a modula-2 program and parses the   '
'   program into operands and operators. From these      '
'   operands and operators, Halstead's and McCabe's      '
'   metrics are calculated.                              '
' Author: Budy Tjahjo                                    '
' Update: Dec 10, 1988                                   '
'                                                        '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'BEGIN
        DEFINT A-Z
        TYPE symbol
             word AS STRING * 15
             wordType AS INTEGER
             count AS INTEGER
        END TYPE

        CONST EOFILE = -1
        CONST OPERATOR = 1
        CONST OPERAND = 2

        DIM SHARED symTab(500) AS symbol       ' symbol table
        DIM SHARED nTable                      ' number of keywords in the program
        DIM SHARED delim$(30)                  ' delimeters
        DIM SHARED key$(50)                    ' keywords
        DIM SHARED lastEnt                     ' last record number
        DIM SHARED commentFlag                 ' comment toggle

        CALL loadKey                                         ' load keywords
        DO
            CLS
            INPUT "Enter the filename "; fileName$           ' get the file
            CALL strip(fileName$)
            IF LEN(fileName$) = 0 THEN EXIT DO               ' no more file
            CALL initialize(fileName$)                       ' prepare the file

            CALL readString(rec$, mode)                      ' read the first record
            WHILE (mode <> EOFILE)                           ' loop until eof
                  CALL strip(rec$)
                  IF LEN(rec$) > 0 THEN CALL insert(rec$)    ' parse and insert
                  CALL readString(rec$, mode)                ' read the next record
            WEND
            CALL clearAll
        LOOP
        LPRINT "******* H a l s t e a d ******** "
        CALL halstead

        LPRINT "******* McCabe complexity *********"
        CALL mcCabe
END
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    close the file when finish processing          '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB clearAll
       CLOSE #1
END SUB


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   get the position of delimeter del$ inside word$     '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB getDelimPos (word$, del$, post)
     post = 0
     DO
           index = index + 1
           IF index > 29 THEN post = 0: EXIT SUB
           post = INSTR(word$, delim$(index))
     LOOP UNTIL post > 0
     del$ = delim$(index)
END SUB

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  calculate hastead metrics from the symbol table      '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB halstead
      FOR i = 1 TO nTable
          IF symTab(i).wordType = OPERAND THEN
                  smallN1 = smallN1 + 1
                  bigN1 = bigN1 + symTab(i).count
          ELSE
                  smallN2 = smallN2 + 1
                  bigN2 = bigN2 + symTab(i).count
          END IF
      NEXT
      LPRINT "number of unique operators : "; smallN1
      LPRINT "number of unique operands : "; smallN2
      LPRINT "total occurences of operators : "; bigN1
      LPRINT "total occurences of operands : "; bigN2
      smallN = smallN1 + smallN2
      bigN = bigN1 + bigN2
      V# = bigN * LOG(smallN)
      LPRINT "v : "; V#
      impLevel# = (2 * smallN2) / (smallN1 * bigN2)
      impEffort# = V# / impLevel#
      langLevel# = (impLevel# * impLevel#) * V#
      LPRINT "Level of Implementation of a program : "; impLevel#
      LPRINT "The difficulty :"; 1 / impLevel#
      LPRINT "Implementation Effort : "; impEffort#
      LPRINT "Level of programming language : "; langLevel#
      LPRINT : LPRINT
END SUB
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   get the file ready                                   '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB initialize (fileName$)
    OPEN "i", #1, fileName$
END SUB

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   parse the rec$ and insert the keywords into symbol   '
'   table                                                '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB insert (rec$)
        DIM word$(50)

        CALL split(rec$, word$(), nword)
        FOR i = 1 TO nword
            wordType = isKeyWord(word$(i))
            IF wordType = OPERATOR THEN
                  IF word$(i) = "(*" THEN
                        commentFlag = -1
                  ELSEIF word$(i) = "*)" THEN
                        commentFlag = 0
                  ELSE
                        CALL insert2SymbTab(word$(i), OPERATOR)
                  END IF
            ELSE
                  CALL insert2SymbTab(word$(i), OPERAND)
            END IF
        NEXT
        EXIT SUB
END SUB

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  insert keyword w$ and its type into symbol table      '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB insert2SymbTab (w$, wordType)
        IF commentFlag THEN EXIT SUB
        index = lookUp(w$)
        IF index > nTable THEN
            nTable = index
            symTab(index).word = UCASE$(w$)
            symTab(index).wordType = wordType
        END IF
        symTab(index).count = symTab(index).count + 1
END SUB
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   return the type of word$                           '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
FUNCTION isKeyWord (word$)
      FOR index = 1 TO 47
           IF UCASE$(word$) = key$(index) THEN
                  isKeyWord = OPERATOR
                  EXIT FUNCTION
           END IF
      NEXT
      FOR index = 1 TO 29
           IF word$ = delim$(index) THEN
                  isKeyWord = OPERATOR
                  EXIT FUNCTION
           END IF
      NEXT
      isKeyWord = OPERAND
END FUNCTION


''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  load modula-2 keyword$ and delimeters               '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB loadKey
            key$(1) = "AND"
            key$(2) = "ARRAY"
            key$(3) = "BEGIN"
            key$(4) = "BITSET"
            key$(5) = "BOOLEAN"
            key$(6) = "BY"
            key$(7) = "CASE"
            key$(8) = "CARDINAL"
            key$(9) = "CHAR"
            key$(10) = "CONST"
            key$(11) = "DIV"
            key$(12) = "DO"
            key$(13) = "ELSE"
            key$(14) = "ELSIF"
            key$(15) = "END"
            key$(16) = "EXIT"
            key$(17) = "EXPORT"
            key$(18) = "FALSE"
            key$(19) = "FOR"
            key$(20) = "FROM"
            key$(21) = "IF"
            key$(22) = "IMPORT"
            key$(23) = "IN"
            key$(24) = "INTEGER"
            key$(25) = "LOOP"
            key$(26) = "MOD"
            key$(27) = "MODULE"
            key$(28) = "NOT"
```

```
            key$(29) = "OF"
            key$(30) = "OR"
            key$(31) = "POINTER"
            key$(32) = "PROCEDURE"
            key$(33) = "QUALIFIED"
            key$(34) = "RECORD"
            key$(35) = "REPEAT"
            key$(36) = "RETURN"
            key$(37) = "SET"
            key$(38) = "THEN"
            key$(39) = "TO"
            key$(40) = "TRUE"
            key$(41) = "TYPE"
            key$(42) = "UNTIL"
            key$(43) = "VAR"
            key$(44) = "WHILE"
            key$(45) = "WITH"
            key$(46) = "IMPLEMENTATION"
            key$(47) = "DEFINITION"

            delim$(1) = "(*"
            delim$(2) = "*)"
            delim$(3) = ":="
            delim$(4) = "<>"
            delim$(5) = "<="
            delim$(6) = ">="
            delim$(7) = ".."
            delim$(8) = "+"
            delim$(9) = "-"
            delim$(10) = "*"
            delim$(11) = "/"
            delim$(12) = "&"
            delim$(13) = "."
            delim$(14) = ","
            delim$(15) = "("
            delim$(16) = ")"
            delim$(17) = "["
            delim$(18) = "]"
            delim$(19) = "{"
            delim$(20) = "}"
            delim$(21) = "^"
            delim$(22) = "~"
            delim$(23) = "="
            delim$(24) = "#"
            delim$(25) = "<"
            delim$(26) = ">"
            delim$(27) = ":"
            delim$(28) = "|"
            delim$(29) = ";"
END SUB
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   return the index of word$ inside the symbol table    '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
FUNCTION lookUp (word$)
      index = 1
      WHILE index <= nTable
              temp$ = symTab(index).word
              CALL strip(temp$)
              IF temp$ = UCASE$(word$) THEN
                     lookUp = index
                     EXIT FUNCTION
              ELSE
                     index = index + 1
              END IF
      WEND
      lookUp = index
END FUNCTION


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    measure the McCabe's metric and print it            '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB mcCabe
      FOR i = 1 TO nTable
          key$ = symTab(i).word
          CALL strip(key$)
          IF key$ = "¦" THEN count = count + symTab(i).count
          IF key$ = "&" THEN count = count + symTab(i).count
          IF key$ = "AND" THEN count = count + symTab(i).count
          IF key$ = "OR" THEN count = count + symTab(i).count
          IF key$ = "PROCEDURE" THEN count = count + symTab(i).count
          IF key$ = "CASE" THEN count = count + symTab(i).count
          IF key$ = "FOR" THEN count = count + symTab(i).count
          IF key$ = "MODULE" THEN count = count + symTab(i).count
          IF key$ = "REPEAT" THEN count = count + symTab(i).count
          IF key$ = "WHILE" THEN count = count + symTab(i).count
          IF key$ = "IMPLEMENTATION" THEN count = count + symTab(i).count
          IF key$ = "DEFINITION" THEN count = count + symTab(i).count
      NEXT
      LPRINT "v(G) = "; count + 1

END SUB
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   read record from the file                            '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB readString (line$, mode)

      IF EOF(1) THEN
           mode = EOFILE
      ELSE
           INPUT #1, line$
           mode = 0
      END IF

END SUB


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  split the rec$ into keywords  and insert the keywords'
'  into word$() array. nword is # of keywords inside the'
'   word$()                                              '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB split (rec$, word$(), nword)
       DIM wordLine$(30)

       DO    ' split into word seperated by a blank
            CALL strip(rec$)
            IF LEN(rec$) = 0 THEN EXIT DO
            post = INSTR(rec$, " ")
            nwl = nwl + 1
            IF post = 0 THEN
                 wordLine$(nwl) = rec$
            ELSE
                 wordLine$(nwl) = LEFT$(rec$, post - 1)
                 rec$ = MID$(rec$, post + 1, LEN(rec$))
            END IF
       LOOP UNTIL post = 0

       nword = 0
       FOR i = 1 TO nwl            ' spilt into words seperated by delim
            rec$ = wordLine$(i)
            DO
                 CALL getDelimPos(rec$, del$, post)
                 IF post = 0 THEN
                       nword = nword + 1
                       word$(nword) = rec$
                 ELSE
                       IF post > 1 THEN
                           nword = nword + 1
                           word$(nword) = MID$(rec$, 1, post - 1)
                       END IF
                       nword = nword + 1
                       word$(nword) = del$
                       IF post = LEN(rec$) THEN EXIT DO
```

```
                    rec$ = MID$(rec$, post + 1, LEN(rec$))
                END IF
            LOOP UNTIL post = 0
        NEXT
END SUB


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' strip blanks of the word$                                                     '
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
SUB strip (word$)
     word$ = RTRIM$(word$)
     word$ = LTRIM$(word$)
END SUB
```

APPENDIX H

SAS TABLES AND PLOTS

The following tables and plots are for the expert programs analysis. The variables are represented as

l_m.

Where,

l could be A for Ada; or
C for C++; or
M for Modula-2;

and

m is the selected metrics. (see nomenclature).

OBS	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VG	A_HK	A_TSO	A_DH	A_DA	A_N	A_NE
1	101	76	330	603	4829	401	1936426	0.015	35	86.67	2	5	2	933	795
2	26	24	192	257	1756	139	244519	0.091	1	1.00	1	8	2	449	161
3	17	26	109	172	1067	56	59430	0.334	8	1.00	2	4	2	261	133
4	28	28	191	292	1944	146	283859	0.009	3	1.78	1	10	3	483	187
5	29	30	112	214	1329	103	137492	0.124	3	30.00	2	6	2	326	200
6	16	31	50	113	628	29	18301	0.738	4	10.50	1	4	2	163	151
7	18	32	67	159	884	45	39537	0.442	4	18.00	1	4	2	226	163
8	355	265	1132	2142	21051	1435	30208185	0.010	78	81.60	13	49	2	3385	3083

OBS	C_N1	C_N2	C_BN1	C_BN2	C_V	C_D	C_E	C_L	C_VG	C_HK	C_TSO	C_DH	C_DA	C_N	C_NE
1	70	39	376	633	4734	568	2688912	0.015	24	39.37	1	11	7	1009	440
2	30	25	156	198	1426	119	169481	0.101	1	1.00	1	6	2	356	183
3	17	23	101	170	1000	63	62906	0.253	6	1.00	1	3	1	271	120
4	57	41	297	493	3622	343	1241284	0.003	9	5.66	1	13	2	790	383
5	36	36	143	269	1773	142	251756	0.066	9	11.25	1	6	2	412	267
6	52	53	160	314	2205	154	339570	0.093	9	12.00	1	9	3	474	416
7	51	53	159	311	3713	150	29550	0.165	9	12.00	1	10	3	470	411
8	246	119	911	1558	14567	1610	23452870	0.006	48	50.36	3	20	3	2469	1923

OBS	M_N1	M_N2	M_BN1	M_BN2	M_V	M_D	M_E	M_L	M_VG	M_HK	M_TSO	M_DH	M_DA	M_N	M_NE
1	65	48	311	521	3933	353	1387473	0.032	22	199.88	2	6	3	832	457
2	26	23	170	234	1572	132	207953	0.013	1	1.00	1	6	1	404	157
3	19	26	99	160	1062	66	69850	0.246	8	1.00	2	4	2	279	141
4	30	32	200	320	2321	157	473758	0.005	5	2.34	1	10	3	520	213
5	19	27	104	193	1137	68	77217	0.247	3	11.25	2	5	2	297	145
6	26	31	107	215	1301	90	117378	0.160	5	12.00	3	9	2	322	191
7	32	31	151	281	1790	145	259583	0.085	6	12.86	3	7	2	432	217
8	335	212	1137	2228	21214	1760	37336640	0.007	56	136.69	6	40	3	3365	3083

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
A_N1	8	73.750000	117.03693	590.00000	16.00000000	355.00000
A_N2	8	64.000000	82.94060	512.00000	24.00000000	265.00000
A_BN1	8	272.875000	358.44842	2183.00000	50.00000000	1132.00000
A_BN2	8	494.000000	682.87501	3952.00000	113.00000000	2142.00000
A_V	8	4186.000000	6940.83482	33488.00000	628.00000000	21051.00000
A_D	8	294.250000	475.78649	2354.00000	29.00000000	1435.00000
A_E	8	4115968.625000	10562192.03724	32927749.00000	18301.00000000	30208185.00000
A_L	8	2.203750000E-01	2.637926337E-01	1.76300	9.000000000E-03	7.380000000E-01
A_VG	8	17.000000	27.02380	136.00000	1.000000000E+00	78.00000
A_HK	8	28.818750	35.58793	230.55000	1.000000000E+00	86.67000
A_TSO	8	2.875000	4.12094	23.00000	1.000000000E+00	13.00000
A_DH	8	11.250000	15.40640	90.00000	4.00000000	49.00000
A_DA	8	2.125000	3.535533906E-01	17.00000	2.00000000	3.00000
A_N	8	778.250000	1072.12063	6226.00000	163.00000000	3365.00000
A_NE	8	609.125000	1023.76036	4873.00000	133.00000000	3083.00000
C_N1	8	70.125000	72.96073	561.00000	17.00000000	246.00000
C_N2	8	48.625000	30.50966	389.00000	23.00000000	119.00000
C_BN1	8	288.125000	267.76132	2305.00000	101.00000000	911.00000
C_BN2	8	483.250000	456.77745	3946.00000	170.00000000	1558.00000
C_V	8	4130.000000	4409.36225	33040.00000	1000.00000000	14567.00000
C_D	8	393.625000	517.87448	3149.00000	63.00000000	1610.00000
C_E	8	3528628.625000	6100791.82114	28238229.00000	28550.00000000	23452870.00000
C_L	8	8.050000000E-02	8.666520145E-02	7.240000000E-01	3.000000000E-03	2.530000000E-01
C_VG	8	14.625000	14.91823	117.00000	1.000000000E+00	48.00000
C_HK	8	16.550000	18.28610	132.64000	1.000000000E+00	50.36000
C_TSO	8	1.250000	7.071067812E-01	10.00000	1.000000000E+00	3.00000
C_DH	8	10.000000	6.07093	80.00000	3.00000000	20.00000
C_DA	8	2.875000	1.80772	23.00000	1.000000000E+00	7.00000
C_N	8	761.375000	724.00314	6251.00000	271.00000000	2469.00000
C_NE	8	517.875000	579.87691	4143.00000	120.00000000	1923.00000
M_N1	8	69.000000	108.46988	552.00000	19.00000000	335.00000
M_N2	8	53.750000	64.38223	430.00000	23.00000000	212.00000
M_BN1	8	284.875000	351.32848	2279.00000	99.00000000	1137.00000
M_BN2	8	521.500000	698.13200	4172.00000	190.00000000	2228.00000
M_V	8	4291.250000	6900.92345	34330.00000	1062.00000000	21214.00000
M_D	8	346.375000	578.49335	2771.00000	66.00000000	1760.00000
M_E	8	4901231.625000	13076753.18591	39209853.00000	69850.00000000	37336640.00000
M_L	8	9.937500000E-02	1.046178858E-01	7.950000000E-01	5.000000000E-03	2.470000000E-01
M_VG	8	13.250000	16.42165	106.00000	1.000000000E+00	56.00000
M_HK	8	47.126250	76.82056	377.01000	1.000000000E+00	199.88000
M_TSO	8	2.500000	1.60357	20.00000	1.000000000E+00	6.00000
M_DH	8	10.875000	11.93360	87.00000	4.00000000	40.00000
M_DA	8	2.250000	7.071067812E-01	18.00000	1.000000000E+00	3.00000
M_N	8	806.375000	1049.04704	6451.00000	278.00000000	3365.00000
M_NE	8	575.500000	1018.30602	4604.00000	141.00000000	3083.00000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VG	A_HK	A_TSO	A_OH	A_DA
A_N1	1.00000 0.0000	0.99755 0.0001	0.99312 0.0001	0.99832 0.0001	0.99780 0.0001	0.99857 0.0001	0.98361 0.0001	-0.43172 0.2855	0.97852 0.0001	0.76082 0.0285	0.97768 0.0001	0.96092 0.0001	-0.15795 0.7087
A_N2	0.99755 0.0001	1.00000 0.0000	0.98598 0.0001	0.99468 0.0001	0.99615 0.0001	0.99337 0.0001	0.98924 0.0001	-0.37335 0.3623	0.97415 0.0001	0.74326 0.0346	0.96306 0.0001	0.96292 0.0001	-0.17538 0.6778
A_BN1	0.99312 0.0001	0.98598 0.0001	1.00000 0.0000	0.99769 0.0001	0.99621 0.0001	0.99771 0.0001	0.99016 0.0001	-0.49677 0.2105	0.96121 0.0001	0.71441 0.0465	0.97117 0.0001	0.97218 0.0001	-0.09229 0.8279
A_BN2	0.99832 0.0001	0.99468 0.0001	0.99769 0.0001	1.00000 0.0000	0.99941 0.0001	0.99949 0.0001	0.99638 0.0001	-0.45742 0.2545	0.96949 0.0001	0.73230 0.0388	0.97890 0.0001	0.97155 0.0001	-0.11952 0.7780
A_V	0.99780 0.0001	0.99615 0.0001	0.99621 0.0001	0.99941 0.0001	1.00000 0.0000	0.99813 0.0001	0.99129 0.0001	-0.43259 0.2844	0.96564 0.0001	0.71707 0.0453	0.98416 0.0001	0.97615 0.0001	-0.13052 0.7580
A_D	0.99857 0.0001	0.99337 0.0001	0.99771 0.0001	0.99949 0.0001	0.99813 0.0001	1.00000 0.0000	0.99164 0.0001	-0.46857 0.2416	0.97305 0.0001	0.74811 0.0328	0.97387 0.0001	0.96563 0.0001	-0.12590 0.7664
A_E	0.98361 0.0001	0.98924 0.0001	0.99016 0.0001	0.99638 0.0001	0.99129 0.0001	0.99164 0.0001	1.00000 0.0000	-0.35033 0.3949	0.93416 0.0007	0.64218 0.0860	0.98412 0.0001	0.98611 0.0001	-0.14660 0.7290
A_L	-0.43172 0.2855	-0.37335 0.3623	-0.49677 0.2105	-0.45742 0.2545	-0.43259 0.2844	-0.46857 0.2416	-0.35033 0.3949	1.00000 0.0000	-0.41651 0.3047	-0.43884 0.2770	-0.35648 0.3861	-0.40954 0.3137	-0.32377 0.4340
A_VG	0.97852 0.0001	0.97415 0.0001	0.96121 0.0001	0.96949 0.0001	0.96564 0.0001	0.97305 0.0001	0.93416 0.0007	-0.41651 0.3047	1.00000 0.0000	0.84629 0.0081	0.93260 0.0007	0.88767 0.0033	-0.20933 0.6188
A_HK	0.76082 0.0285	0.74326 0.0346	0.71441 0.0465	0.73230 0.0388	0.71707 0.0453	0.74811 0.0328	0.64218 0.0860	-0.43864 0.2770	0.84629 0.0081	1.00000 0.0000	0.64738 0.0827	0.56328 0.1460	-0.30699 0.4595
A_TSO	0.97768 0.0001	0.96306 0.0001	0.97117 0.0001	0.97890 0.0001	0.98416 0.0001	0.97387 0.0001	0.98412 0.0001	-0.35648 0.3861	0.93260 0.0007	0.64738 0.0827	1.00000 0.0000	0.97711 0.0001	-0.18385 0.6630
A_OH	0.96092 0.0001	0.96292 0.0001	0.97218 0.0001	0.97155 0.0001	0.97615 0.0001	0.96563 0.0001	0.98611 0.0001	-0.40954 0.3137	0.88767 0.0033	0.56328 0.1460	0.97711 0.0001	1.00000 0.0000	-0.03278 0.9386
A_DA	-0.15795 0.7087	-0.17538 0.6778	-0.09229 0.8279	-0.11952 0.7780	-0.13052 0.7580	-0.12590 0.7664	-0.14660 0.7290	-0.32377 0.4340	-0.20933 0.6188	-0.30699 0.4595	-0.18385 0.6630	-0.03278 0.9386	1.00000 0.0000
A_N	0.99704 0.0001	0.99288 0.0001	0.99886 0.0001	0.99978 0.0001	0.99909 0.0001	0.99926 0.0001	0.98592 0.0001	-0.46711 0.2432	0.98624 0.0001	0.72326 0.0428	0.97799 0.0001	0.97356 0.0001	-0.11127 0.7931
A_NE	0.99939 0.0001	0.99936 0.0001	0.99059 0.0001	0.99731 0.0001	0.99786 0.0001	0.99672 0.0001	0.98757 0.0001	-0.40313 0.3220	0.97645 0.0001	0.74966 0.0322	0.98155 0.0001	0.96340 0.0001	-0.16661 0.6933
C_N1	0.97941 0.0001	0.98556 0.0001	0.96828 0.0001	0.97890 0.0001	0.98117 0.0001	0.97601 0.0001	0.98073 0.0001	-0.33824 0.4125	0.93830 0.0006	0.70015 0.0532	0.98221 0.0001	0.96725 0.0001	-0.07269 0.8642
C_N2	0.90138 0.0022	0.92404 0.0010	0.87261 0.0047	0.89634 0.0026	0.90477 0.0020	0.88886 0.0032	0.92832 0.0008	-0.11752 0.7817	0.84867 0.0077	0.60411 0.1127	0.90515 0.0020	0.90926 0.0017	-0.10096 0.8119

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VQ	A_HK	A_TSO	A_DH	A_DA
C_BN1	0.99054 0.0001	0.97482 0.0001	0.98102 0.0001	0.98293 0.0001	0.97915 0.0001	0.98378 0.0001	0.95561 0.0002	-0.47800 0.2309	0.95940 0.0002	0.75135 0.0316	0.93606 0.0006	0.94747 0.0003	0.01339 0.9749
C_BN2	0.97876 0.0001	0.97726 0.0001	0.97124 0.0001	0.97835 0.0001	0.97569 0.0001	0.97830 0.0001	0.95647 0.0002	-0.43006 0.2875	0.95989 0.0002	0.76005 0.0286	0.93874 0.0005	0.94238 0.0005	-0.00022 0.9996
C_V	0.97490 0.0001	0.97948 0.0001	0.96428 0.0001	0.97500 0.0001	0.97483 0.0001	0.97216 0.0001	0.96650 0.0001	-0.37826 0.3555	0.94655 0.0004	0.72725 0.0409	0.94627 0.0004	0.94988 0.0003	-0.04655 0.9128
C_D	0.98829 0.0001	0.98518 0.0001	0.98472 0.0001	0.98919 0.0001	0.98621 0.0001	0.98899 0.0001	0.96456 0.0001	-0.45679 0.2552	0.97199 0.0001	0.76811 0.0257	0.94978 0.0003	0.94907 0.0003	-0.03950 0.9260
C_E	0.99872 0.0001	0.99331 0.0001	0.99665 0.0001	0.99225 0.0001	0.99558 0.0001	0.99852 0.0001	0.99826 0.0001	-0.37472 0.3604	0.94626 0.0004	0.66756 0.0705	0.99065 0.0001	0.98667 0.0001	-0.11414 0.7878
C_L	-0.50322 0.2037	-0.47136 0.2384	-0.51634 0.1882	-0.50651 0.2002	-0.48865 0.2192	-0.52169 0.1848	-0.42274 0.2967	0.54320 0.1641	-0.49096 0.2276	-0.56181 0.1473	-0.37180 0.3645	-0.46713 0.2432	-0.40795 0.3157
C_VQ	0.96681 0.0001	0.96718 0.0001	0.93967 0.0005	0.95532 0.0002	0.95188 0.0003	0.95744 0.0002	0.92470 0.0010	-0.37622 0.3583	0.96723 0.0001	0.85476 0.0068	0.92398 0.0010	0.87749 0.0042	-0.15235 0.7187
C_HK	0.86823 0.0061	0.86285 0.0058	0.82567 0.0115	0.84568 0.0082	0.83601 0.0097	0.85629 0.0066	0.78173 0.0218	-0.35890 0.3826	0.92809 0.0009	0.96480 0.0001	0.77286 0.0245	0.71292 0.0471	-0.24156 0.5644
C_TSO	0.97099 0.0001	0.97921 0.0001	0.96845 0.0001	0.97513 0.0001	0.98180 0.0001	0.96878 0.0001	0.99817 0.0001	-0.32224 0.4363	0.91207 0.0016	0.59927 0.1164	0.99276 0.0001	0.99006 0.0001	-0.14286 0.7358
C_DH	0.82611 0.0118	0.82504 0.0117	0.83136 0.0105	0.83384 0.0101	0.82900 0.0110	0.83357 0.0101	0.80923 0.0150	-0.38574 0.3453	0.78186 0.0219	0.60836 0.1095	0.75199 0.0314	0.82835 0.0111	0.23905 0.5686
C_DA	0.24386 0.5610	0.22286 0.5856	0.19398 0.6453	0.20819 0.6208	0.18603 0.6592	0.23224 0.5799	0.08288 0.8453	-0.18323 0.6641	0.39186 0.3370	0.75660 0.0299	0.06513 0.8868	-0.00385 0.9928	-0.19558 0.6425
C_N	0.99014 0.0001	0.97708 0.0001	0.97557 0.0001	0.98077 0.0001	0.97769 0.0001	0.98106 0.0001	0.95686 0.0002	-0.44811 0.2655	0.96042 0.0002	0.75740 0.0295	0.93844 0.0006	0.94496 0.0004	0.00481 0.9910
C_NE	0.97012 0.0001	0.98072 0.0001	0.95610 0.0002	0.96928 0.0001	0.97383 0.0001	0.96464 0.0001	0.88222 0.0001	-0.28505 0.4938	0.92406 0.0010	0.66744 0.0705	0.96457 0.0001	0.96746 0.0001	-0.09398 0.8248
M_N1	0.99209 0.0001	0.98633 0.0001	0.98619 0.0001	0.99302 0.0001	0.99607 0.0001	0.98980 0.0001	0.99678 0.0001	-0.36273 0.3772	0.95415 0.0002	0.68756 0.0595	0.98690 0.0001	0.97966 0.0001	-0.14528 0.7314
M_N2	0.98906 0.0001	0.99507 0.0001	0.98116 0.0001	0.98963 0.0001	0.99350 0.0001	0.98561 0.0001	0.99766 0.0001	-0.34427 0.4037	0.94934 0.0003	0.67754 0.0649	0.99006 0.0001	0.98059 0.0001	-0.13650 0.7472
M_BN1	0.99406 0.0001	0.98329 0.0001	0.99438 0.0001	0.99716 0.0001	0.99761 0.0001	0.99538 0.0001	0.98920 0.0001	-0.42738 0.2909	0.95899 0.0002	0.70431 0.0511	0.97535 0.0001	0.97762 0.0001	-0.09761 0.8181
M_BN2	0.99322 0.0001	0.99602 0.0001	0.98901 0.0001	0.99487 0.0001	0.99703 0.0001	0.99202 0.0001	0.99466 0.0001	-0.38647 0.3443	0.95630 0.0002	0.68488 0.0658	0.98355 0.0001	0.97989 0.0001	-0.11662 0.7833
M_V	0.99182 0.0001	0.99475 0.0001	0.98858 0.0001	0.99408 0.0001	0.99669 0.0001	0.99070 0.0001	0.99664 0.0001	-0.38322 0.3487	0.95108 0.0003	0.68020 0.0634	0.98625 0.0001	0.98378 0.0001	-0.11536 0.7856

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VG	A_HK	A_TSO	A_DH	A_DA
M_D	0.99304 0.0001	0.99634 0.0001	0.99002 0.0001	0.99549 0.0001	0.99754 0.0001	0.99285 0.0001	0.99465 0.0001	-0.38696 0.3436	0.95770 0.0002	0.69606 0.0648	0.98345 0.0001	0.97887 0.0001	-0.13227 0.7549
M_E	0.97909 0.0001	0.98503 0.0001	0.97518 0.0001	0.98165 0.0001	0.98733 0.0001	0.97607 0.0001	0.99057 0.0001	-0.33743 0.4137	0.92410 0.0010	0.62178 0.0986	0.99334 0.0001	0.98980 0.0001	-0.13959 0.7417
M_L	-0.43171 0.2858	-0.40582 0.3184	-0.48315 0.2252	-0.45367 0.2589	-0.43815 0.2776	-0.46158 0.2496	-0.38000 0.3531	0.49188 0.2157	-0.41091 0.3119	-0.33299 0.4202	-0.29214 0.4826	-0.42905 0.2888	-0.36450 0.3747
M_VG	0.96472 0.0001	0.96520 0.0001	0.96760 0.0001	0.97786 0.0001	0.97630 0.0001	0.97847 0.0001	0.95509 0.0002	-0.38063 0.3523	0.99548 0.0001	0.80407 0.0161	0.95267 0.0003	0.91283 0.0015	-0.18096 0.6680
M_HK	0.66356 0.0728	0.63943 0.0878	0.62817 0.0947	0.63716 0.0893	0.61771 0.1027	0.65853 0.0770	0.52245 0.1841	-0.43945 0.2760	0.78427 0.0212	0.95693 0.0002	0.51687 0.1896	0.44150 0.2735	-0.23557 0.5744
M_TSO	0.84111 0.0089	0.87840 0.0044	0.76798 0.0202	0.82385 0.0118	0.83778 0.0094	0.81488 0.0137	0.87633 0.0043	0.08068 0.8496	0.81756 0.0132	0.58289 0.1214	0.87553 0.0044	0.81533 0.0136	-0.37796 0.3559
M_DH	0.98214 0.0003	0.98255 0.0001	0.98170 0.0003	0.98538 0.0002	0.98609 0.0001	0.98145 0.0003	0.98324 0.0001	-0.28636 0.4885	0.98507 0.0035	0.96348 0.1458	0.98407 0.0001	0.98623 0.0001	-0.02963 0.9445
M_DA	0.53599 0.1709	0.52858 0.1780	0.51586 0.1907	0.53017 0.1765	0.51509 0.1914	0.53269 0.1741	0.45815 0.2536	-0.38121 0.3515	0.60556 0.1116	0.83248 0.0824	0.45348 0.2581	0.43930 0.2762	0.42857 0.2894
M_N	0.99388 0.0001	0.99550 0.0001	0.99120 0.0001	0.99610 0.0001	0.99761 0.0001	0.99354 0.0001	0.99323 0.0001	-0.40032 0.3257	0.95758 0.0002	0.69818 0.0841	0.99120 0.0001	0.97952 0.0001	-0.11030 0.7949
M_NE	0.98853 0.0001	0.99403 0.0001	0.98274 0.0001	0.98998 0.0001	0.99402 0.0001	0.98591 0.0001	0.98890 0.0001	-0.35034 0.3949	0.94582 0.0004	0.68793 0.0703	0.99087 0.0001	0.98352 0.0001	-0.14384 0.7340

	A_N	A_NE	C_N1	C_N2	C_BN1	C_BN2	C_V	C_D	C_E	C_L	C_VG	C_HK	C_TSO
A_N1	0.99704 0.0001	0.99838 0.0001	0.97941 0.0001	0.90136 0.0022	0.98054 0.0001	0.97876 0.0001	0.97490 0.0001	0.98929 0.0001	0.98972 0.0001	-0.50322 0.2037	0.96681 0.0001	0.66923 0.0051	0.97099 0.0001
A_N2	0.99258 0.0001	0.99836 0.0001	0.96556 0.0001	0.92404 0.0010	0.97482 0.0001	0.97728 0.0001	0.97948 0.0001	0.99518 0.0001	0.99331 0.0001	-0.47136 0.2384	0.96719 0.0001	0.88285 0.0056	0.97921 0.0001
A_BN1	0.99886 0.0001	0.99058 0.0001	0.96828 0.0001	0.87261 0.0047	0.98102 0.0001	0.97124 0.0001	0.96428 0.0001	0.98472 0.0001	0.98665 0.0001	-0.51834 0.1882	0.93967 0.0005	0.82567 0.0115	0.96845 0.0001
A_BN2	0.99978 0.0001	0.99731 0.0001	0.97890 0.0001	0.89634 0.0026	0.98293 0.0001	0.97835 0.0001	0.97500 0.0001	0.99919 0.0001	0.99225 0.0001	-0.50651 0.2002	0.95532 0.0002	0.84568 0.0082	0.97513 0.0001
A_V	0.99909 0.0001	0.99786 0.0001	0.98117 0.0001	0.90477 0.0020	0.97915 0.0001	0.97569 0.0001	0.97483 0.0001	0.98621 0.0001	0.99558 0.0001	-0.48965 0.2192	0.95188 0.0003	0.83601 0.0097	0.98180 0.0001
A_D	0.99928 0.0001	0.99672 0.0001	0.97601 0.0001	0.88666 0.0032	0.98378 0.0001	0.97630 0.0001	0.97216 0.0001	0.98999 0.0001	0.98852 0.0001	-0.52169 0.1848	0.95744 0.0002	0.85629 0.0066	0.96878 0.0001
A_E	0.98592 0.0001	0.98767 0.0001	0.98073 0.0001	0.92932 0.0008	0.95561 0.0002	0.95647 0.0002	0.96650 0.0001	0.96456 0.0001	0.99826 0.0001	-0.42274 0.2967	0.92470 0.0010	0.78173 0.0219	0.99817 0.0001

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	A_N	A_NE	C_N1	C_N2	C_BN1	C_BN2	C_V	C_D	C_E	C_L	C_VG	C_HK	C_TSO
A_L	-0.46711 0.2432	-0.40313 0.3220	-0.33824 0.4125	-0.11752 0.7817	-0.47800 0.2309	-0.43006 0.2875	-0.37826 0.3555	-0.45679 0.2552	-0.37472 0.3604	0.54320 0.1641	-0.37622 0.3583	-0.35890 0.3826	-0.32224 0.4363
A_VG	0.96624 0.0001	0.97645 0.0001	0.93930 0.0005	0.84867 0.0077	0.95940 0.0002	0.95988 0.0002	0.94655 0.0004	0.97199 0.0001	0.94626 0.0004	-0.48096 0.2276	0.98723 0.0001	0.92809 0.0009	0.91207 0.0016
A_HK	0.72326 0.0426	0.74966 0.0322	0.70015 0.0532	0.60411 0.1127	0.75135 0.0316	0.76005 0.0286	0.72725 0.0409	0.76911 0.0257	0.66756 0.0705	-0.56161 0.1473	0.85476 0.0068	0.96490 0.0001	0.59927 0.1164
A_TSO	0.97798 0.0001	0.99155 0.0001	0.96221 0.0001	0.90515 0.0020	0.93606 0.0006	0.93874 0.0005	0.94627 0.0004	0.94978 0.0003	0.99065 0.0001	-0.37180 0.3646	0.92398 0.0010	0.77288 0.0245	0.99276 0.0001
A_DH	0.97356 0.0001	0.96340 0.0001	0.96725 0.0001	0.90926 0.0017	0.94747 0.0003	0.94238 0.0005	0.94988 0.0003	0.94907 0.0003	0.98667 0.0001	-0.46713 0.2432	0.87749 0.0042	0.71292 0.0471	0.99006 0.0001
A_DA	-0.11127 0.7931	-0.16661 0.6933	-0.07269 0.8642	-0.10098 0.8119	0.01339 0.9749	-0.00022 0.9996	-0.04655 0.9128	-0.03950 0.9260	-0.11414 0.7878	-0.40795 0.3157	-0.15235 0.7187	-0.24156 0.5644	-0.14286 0.7358
A_N	1.00000 0.0000	0.99572 0.0001	0.97646 0.0001	0.89063 0.0030	0.98226 0.0001	0.97814 0.0001	0.97211 0.0001	0.98776 0.0001	0.99169 0.0001	-0.50774 0.1990	0.94977 0.0003	0.83719 0.0095	0.97489 0.0001
A_NE	0.99572 0.0001	1.00000 0.0000	0.98279 0.0001	0.91257 0.0016	0.97797 0.0001	0.97804 0.0001	0.97727 0.0001	0.98749 0.0001	0.99250 0.0001	-0.48557 0.2225	0.96653 0.0001	0.86428 0.0056	0.97640 0.0001
C_N1	0.97646 0.0001	0.98279 0.0001	1.00000 0.0000	0.96460 0.0001	0.98014 0.0001	0.98679 0.0001	0.98271 0.0001	0.98393 0.0001	0.98693 0.0001	-0.54556 0.1619	0.94911 0.0003	0.84427 0.0084	0.97401 0.0001
C_N2	0.89063 0.0030	0.91257 0.0016	0.96460 0.0001	1.00000 0.0000	0.90054 0.0023	0.92357 0.0011	0.95165 0.0003	0.90716 0.0019	0.92943 0.0008	-0.46445 0.2463	0.88350 0.0036	0.77818 0.0229	0.93203 0.0007
C_BN1	0.98226 0.0001	0.97797 0.0001	0.98014 0.0001	0.90054 0.0023	1.00000 0.0000	0.99683 0.0001	0.98367 0.0001	0.99808 0.0001	0.97065 0.0001	-0.62421 0.0981	0.95983 0.0002	0.87161 0.0048	0.93994 0.0005
C_BN2	0.97814 0.0001	0.97804 0.0001	0.98679 0.0001	0.92357 0.0011	0.99683 0.0001	1.00000 0.0000	0.98996 0.0001	0.99721 0.0001	0.97112 0.0001	-0.61202 0.1069	0.97079 0.0001	0.86421 0.0036	0.94167 0.0005
C_V	0.97211 0.0001	0.97727 0.0001	0.98271 0.0001	0.95165 0.0003	0.98367 0.0001	0.98996 0.0001	1.00000 0.0000	0.98594 0.0001	0.97528 0.0001	-0.54209 0.1651	0.95745 0.0002	0.86212 0.0059	0.95642 0.0002
C_D	0.98776 0.0001	0.98749 0.0001	0.98393 0.0001	0.90716 0.0019	0.99808 0.0001	0.99721 0.0001	0.98594 0.0001	1.00000 0.0000	0.97780 0.0001	-0.58283 0.1214	0.97229 0.0001	0.86497 0.0035	0.94905 0.0003
C_E	0.99169 0.0001	0.99250 0.0001	0.98693 0.0001	0.92943 0.0008	0.97065 0.0001	0.97112 0.0001	0.97528 0.0001	0.97780 0.0001	1.00000 0.0000	-0.46374 0.2471	0.93948 0.0005	0.80452 0.0160	0.99376 0.0001
C_L	-0.50774 0.1990	-0.48557 0.2225	-0.54556 0.1619	-0.46445 0.2463	-0.62421 0.0981	-0.61202 0.1069	-0.54209 0.1651	-0.58283 0.1214	-0.46374 0.2471	1.00000 0.0000	-0.51651 0.1900	-0.60049 0.1155	-0.39397 0.3342
C_VG	0.94977 0.0003	0.96653 0.0001	0.94911 0.0003	0.88350 0.0036	0.95983 0.0002	0.97079 0.0001	0.95745 0.0002	0.97229 0.0001	0.93948 0.0005	-0.51651 0.1900	1.00000 0.0000	0.94338 0.0004	0.90396 0.0021
C_HK	0.83719 0.0095	0.86428 0.0056	0.84427 0.0084	0.77818 0.0229	0.87161 0.0048	0.86421 0.0036	0.86212 0.0059	0.86497 0.0035	0.80452 0.0160	-0.60049 0.1155	0.94338 0.0004	1.00000 0.0000	0.74724 0.0331

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	A_N	A_NE	C_N1	C_N2	C_BN1	C_BN2	C_V	C_D	C_E	C_L	C_VG	C_HK	C_TSO
C_TSO	0.97488 0.0001	0.97640 0.0001	0.97401 0.0001	0.93203 0.0007	0.93994 0.0005	0.94187 0.0005	0.95642 0.0002	0.94905 0.0003	0.99376 0.0001	-0.39397 0.3342	0.90396 0.0021	0.74724 0.0331	1.00000 0.0000
C_DH	0.83287 0.0102	0.82504 0.0117	0.89503 0.0027	0.87166 0.0048	0.90714 0.0019	0.90761 0.0018	0.90802 0.0018	0.88648 0.0034	0.83052 0.0107	-0.75870 0.0291	0.80824 0.0152	0.76182 0.0280	0.79682 0.0179
C_DA	0.19830 0.6376	0.22966 0.5843	0.20701 0.6228	0.14667 0.7289	0.29281 0.4816	0.29848 0.4727	0.25530 0.5417	0.29537 0.4775	0.12191 0.7737	-0.49924 0.2078	0.40061 0.3254	0.66691 0.0708	0.02794 0.9476
C_N	0.87912 0.0001	0.87874 0.0001	0.88606 0.0001	0.81573 0.0014	0.89874 0.0001	0.89957 0.0001	0.98836 0.0001	0.99827 0.0001	0.97167 0.0001	-0.61698 0.1032	0.96745 0.0001	0.88020 0.0039	0.94185 0.0005
C_NE	0.96641 0.0001	0.97581 0.0001	0.99754 0.0001	0.97724 0.0001	0.96465 0.0001	0.97444 0.0001	0.99722 0.0001	0.97020 0.0001	0.98517 0.0001	-0.50261 0.2043	0.93648 0.0006	0.82100 0.0125	0.97909 0.0001
M_N1	0.99195 0.0001	0.99508 0.0001	0.98866 0.0001	0.93388 0.0007	0.97134 0.0001	0.97223 0.0001	0.98043 0.0001	0.97929 0.0001	0.99833 0.0001	-0.45709 0.2549	0.94533 0.0004	0.82211 0.0123	0.99068 0.0001
M_N2	0.98818 0.0001	0.99291 0.0001	0.98941 0.0001	0.94068 0.0005	0.96790 0.0001	0.97140 0.0001	0.97905 0.0001	0.97661 0.0001	0.99892 0.0001	-0.44938 0.2640	0.94586 0.0004	0.81566 0.0136	0.99317 0.0001
M_BN1	0.99700 0.0001	0.99451 0.0001	0.98676 0.0001	0.91641 0.0014	0.98460 0.0001	0.98063 0.0001	0.98391 0.0001	0.98861 0.0001	0.99429 0.0001	-0.50638 0.2004	0.94661 0.0004	0.83221 0.0104	0.98002 0.0001
M_BN2	0.99404 0.0001	0.99543 0.0001	0.99049 0.0001	0.93182 0.0008	0.97852 0.0001	0.97870 0.0001	0.98484 0.0001	0.98483 0.0001	0.99795 0.0001	-0.47908 0.2297	0.94902 0.0003	0.82777 0.0112	0.98768 0.0001
M_V	0.99342 0.0001	0.99410 0.0001	0.98915 0.0001	0.93105 0.0008	0.97520 0.0001	0.97505 0.0001	0.98169 0.0001	0.98157 0.0001	0.99900 0.0001	-0.47051 0.2394	0.94304 0.0004	0.81558 0.0136	0.99086 0.0001
M_D	0.99469 0.0001	0.99599 0.0001	0.98847 0.0001	0.92778 0.0009	0.97670 0.0001	0.97591 0.0001	0.98281 0.0001	0.98343 0.0001	0.99742 0.0001	-0.47341 0.2361	0.94728 0.0004	0.82819 0.0111	0.98738 0.0001
M_E	0.88128 0.0001	0.88276 0.0001	0.97903 0.0001	0.93261 0.0007	0.94990 0.0003	0.95136 0.0003	0.96373 0.0001	0.95860 0.0002	0.99679 0.0001	-0.41187 0.3105	0.91581 0.0014	0.76626 0.0266	0.99945 0.0001
M_L	-0.46120 0.2500	-0.41892 0.3016	-0.46582 0.2447	-0.38092 0.3519	-0.54990 0.1579	-0.51118 0.1954	-0.50817 0.1985	-0.50695 0.1998	-0.40673 0.3173	0.67408 0.0668	-0.37244 0.3636	-0.41585 0.3055	-0.35678 0.3856
M_VG	0.87455 0.0001	0.88522 0.0001	0.95805 0.0002	0.88269 0.0037	0.96677 0.0001	0.97035 0.0001	0.96427 0.0001	0.97881 0.0001	0.96485 0.0001	-0.45349 0.2591	0.98857 0.0001	0.90508 0.0020	0.93768 0.0006
M_HK	0.63032 0.0939	0.64844 0.0814	0.58787 0.1254	0.46291 0.2481	0.67764 0.0648	0.67511 0.0662	0.62640 0.0966	0.68781 0.0594	0.55617 0.1523	-0.55172 0.1563	0.77274 0.0246	0.91034 0.0017	0.47104 0.2388
M_TSO	0.81466 0.0136	0.85814 0.0064	0.87853 0.0041	0.93877 0.0005	0.78370 0.0214	0.81778 0.0131	0.86106 0.0060	0.80981 0.0148	0.86496 0.0055	-0.18914 0.6537	0.84499 0.0083	0.74357 0.0345	0.88192 0.0038
M_DH	0.95821 0.0002	0.95842 0.0002	0.88316 0.0001	0.95880 0.0002	0.94866 0.0003	0.95268 0.0003	0.96447 0.0001	0.95016 0.0003	0.98327 0.0001	-0.47841 0.2308	0.88961 0.0031	0.73598 0.0374	0.98615 0.0001
M_DA	0.52302 0.1835	0.53001 0.1767	0.56142 0.1476	0.50823 0.1984	0.64115 0.0867	0.66322 0.0730	0.60650 0.1109	0.62994 0.0941	0.50265 0.2043	-0.60144 0.1147	0.68728 0.0596	0.67723 0.0650	0.42857 0.2894

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	A_N	A_NE	C_N1	C_N2	C_BN1	C_BN2	C_V	C_D	C_E	C_L	C_VG	C_HK	C_TSO
M_N	0.99542 0.0001	0.99552 0.0001	0.99963 0.0001	0.92703 0.0008	0.98094 0.0001	0.97980 0.0001	0.99498 0.0001	0.99648 0.0001	0.99712 0.0001	-0.48841 0.2194	0.94859 0.0003	0.82958 0.0108	0.98550 0.0001
M_NE	0.99599 0.0001	0.99224 0.0001	0.99689 0.0001	0.93608 0.0006	0.96510 0.0001	0.96690 0.0001	0.97610 0.0001	0.97363 0.0001	0.99910 0.0001	-0.44161 0.2733	0.93848 0.0006	0.80604 0.0157	0.99497 0.0001
	C_DH	C_DA	C_N	C_NE	M_N1	M_N2	M_BN1	M_BN2	M_V	M_D	M_E	M_L	M_VG
A_N1	0.82611 0.0116	0.24358 0.5610	0.98014 0.0001	0.97012 0.0001	0.99209 0.0001	0.99806 0.0001	0.99406 0.0001	0.99322 0.0001	0.99162 0.0001	0.99394 0.0001	0.97809 0.0001	-0.43171 0.2855	0.98472 0.0001
A_N2	0.82504 0.0117	0.22296 0.5956	0.97706 0.0001	0.98072 0.0001	0.99633 0.0001	0.99507 0.0001	0.99329 0.0001	0.99602 0.0001	0.99475 0.0001	0.99634 0.0001	0.98503 0.0001	-0.40592 0.3184	0.98520 0.0001
A_BN1	0.83136 0.0105	0.19398 0.6453	0.97557 0.0001	0.95610 0.0002	0.98619 0.0001	0.98116 0.0001	0.99438 0.0001	0.98901 0.0001	0.98858 0.0001	0.99002 0.0001	0.97518 0.0001	-0.48315 0.2252	0.96780 0.0001
A_BN2	0.83384 0.0101	0.20819 0.6208	0.98077 0.0001	0.96928 0.0001	0.99302 0.0001	0.98963 0.0001	0.99716 0.0001	0.99497 0.0001	0.99409 0.0001	0.99549 0.0001	0.98165 0.0001	-0.45367 0.2589	0.97786 0.0001
A_V	0.82900 0.0110	0.19603 0.6592	0.97769 0.0001	0.97383 0.0001	0.99607 0.0001	0.99350 0.0001	0.99761 0.0001	0.99703 0.0001	0.99668 0.0001	0.99754 0.0001	0.98733 0.0001	-0.43815 0.2776	0.97630 0.0001
A_D	0.83357 0.0101	0.23224 0.5799	0.98105 0.0001	0.96464 0.0001	0.98980 0.0001	0.98561 0.0001	0.99538 0.0001	0.99202 0.0001	0.99070 0.0001	0.99285 0.0001	0.97607 0.0001	-0.46159 0.2496	0.97847 0.0001
A_E	0.80823 0.0150	0.08288 0.8453	0.95686 0.0002	0.98222 0.0001	0.99678 0.0001	0.99768 0.0001	0.99820 0.0001	0.99466 0.0001	0.99664 0.0001	0.99465 0.0001	0.99957 0.0001	-0.38000 0.3531	0.95509 0.0002
A_L	-0.38874 0.3453	-0.18323 0.6641	-0.44811 0.2655	-0.28508 0.4938	-0.36273 0.3772	-0.34427 0.4037	-0.42738 0.2909	-0.38647 0.3443	-0.38322 0.3487	-0.38696 0.3436	-0.33743 0.4137	0.49188 0.2157	-0.38063 0.3523
A_VG	0.78186 0.0218	0.39186 0.3370	0.96042 0.0002	0.92406 0.0010	0.95415 0.0002	0.94934 0.0003	0.95899 0.0002	0.95630 0.0002	0.95106 0.0003	0.95770 0.0002	0.92410 0.0010	-0.41091 0.3119	0.99548 0.0001
A_HK	0.80836 0.1095	0.75660 0.0298	0.75740 0.0295	0.66744 0.0705	0.68756 0.0595	0.67754 0.0649	0.70431 0.0511	0.69469 0.0559	0.68020 0.0634	0.69688 0.0548	0.62178 0.0998	-0.33299 0.4203	0.80407 0.0161
A_TSO	0.75109 0.0314	0.05513 0.8968	0.93644 0.0006	0.96457 0.0001	0.98690 0.0001	0.99006 0.0001	0.97535 0.0001	0.98355 0.0001	0.98625 0.0001	0.98345 0.0001	0.99334 0.0001	-0.29214 0.4826	0.95267 0.0003
A_DH	0.82835 0.0111	-0.00385 0.9928	0.94496 0.0004	0.96746 0.0001	0.97966 0.0001	0.98059 0.0001	0.97762 0.0001	0.97989 0.0001	0.98378 0.0001	0.97887 0.0001	0.98980 0.0001	-0.42905 0.2888	0.91283 0.0015
A_DA	0.23906 0.5686	-0.19558 0.6425	0.00481 0.9910	-0.09398 0.8248	-0.14528 0.7314	-0.13650 0.7472	-0.09761 0.8181	-0.11662 0.7833	-0.11538 0.7856	-0.13227 0.7549	-0.13959 0.7417	-0.36450 0.3747	-0.18096 0.6680
A_N	0.83287 0.0102	0.19830 0.6378	0.97912 0.0001	0.96641 0.0001	0.99195 0.0001	0.98818 0.0001	0.99700 0.0001	0.99404 0.0001	0.99342 0.0001	0.99488 0.0001	0.98128 0.0001	-0.46120 0.2500	0.97455 0.0001
A_NE	0.82504 0.0117	0.22966 0.5843	0.97874 0.0001	0.97581 0.0001	0.99508 0.0001	0.99291 0.0001	0.99451 0.0001	0.99543 0.0001	0.99410 0.0001	0.99599 0.0001	0.98276 0.0001	-0.41892 0.3016	0.98522 0.0001

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	C_DH	C_DA	C_N	C_NE	M_N1	M_N2	M_BN1	M_BN2	M_V	M_D	M_E	M_L	M_VG
C_N1	0.88503 0.0027	0.20701 0.6228	0.98506 0.0001	0.99754 0.0001	0.98866 0.0001	0.98941 0.0001	0.98676 0.0001	0.99049 0.0001	0.98915 0.0001	0.98847 0.0001	0.97903 0.0001	-0.46582 0.2447	0.95805 0.0002
C_N2	0.87186 0.0048	0.14867 0.7269	0.91573 0.0014	0.97724 0.0001	0.93388 0.0007	0.94068 0.0005	0.91641 0.0014	0.93182 0.0008	0.93105 0.0008	0.92778 0.0009	0.93261 0.0007	-0.38092 0.3519	0.88269 0.0037
C_BN1	0.90714 0.0018	0.29281 0.4816	0.99874 0.0001	0.96465 0.0001	0.97134 0.0001	0.96790 0.0001	0.98460 0.0001	0.97852 0.0001	0.97520 0.0001	0.97670 0.0001	0.94990 0.0003	-0.54890 0.1579	0.96677 0.0001
C_BN2	0.90761 0.0018	0.29646 0.4727	0.99957 0.0001	0.97444 0.0001	0.97223 0.0001	0.97140 0.0001	0.98083 0.0001	0.97870 0.0001	0.97505 0.0001	0.97591 0.0001	0.95136 0.0003	-0.51118 0.1954	0.97035 0.0001
C_V	0.90902 0.0016	0.25530 0.5417	0.98836 0.0001	0.98722 0.0001	0.98043 0.0001	0.87905 0.0001	0.98391 0.0001	0.98494 0.0001	0.98169 0.0001	0.98261 0.0001	0.96373 0.0001	-0.50817 0.1985	0.96427 0.0001
C_D	0.88646 0.0034	0.28537 0.4775	0.99827 0.0001	0.97020 0.0001	0.97929 0.0001	0.97661 0.0001	0.98861 0.0001	0.98483 0.0001	0.98157 0.0001	0.98343 0.0001	0.95860 0.0002	-0.50695 0.1998	0.97881 0.0001
C_E	0.93052 0.0107	0.12191 0.7737	0.97167 0.0001	0.98517 0.0001	0.99833 0.0001	0.99892 0.0001	0.99429 0.0001	0.99795 0.0001	0.99900 0.0001	0.99742 0.0001	0.99679 0.0001	-0.40673 0.3173	0.96465 0.0001
C_L	-0.75870 0.0281	-0.49824 0.2078	-0.61698 0.1032	-0.50261 0.2043	-0.45709 0.2549	-0.44936 0.2640	-0.50636 0.2004	-0.47906 0.2297	-0.47051 0.2384	-0.47341 0.2361	-0.41197 0.3105	0.67408 0.0668	-0.45349 0.2591
C_VG	0.90824 0.0152	0.40061 0.3254	0.96745 0.0001	0.93648 0.0006	0.94533 0.0004	0.94586 0.0004	0.94861 0.0004	0.94902 0.0003	0.94304 0.0004	0.94728 0.0004	0.91581 0.0014	-0.37244 0.3636	0.98857 0.0001
C_HK	0.76182 0.0280	0.66691 0.0708	0.88020 0.0039	0.82100 0.0125	0.82211 0.0123	0.81566 0.0136	0.83221 0.0104	0.82777 0.0112	0.81558 0.0136	0.82819 0.0111	0.76626 0.0286	-0.41585 0.3065	0.90508 0.0020
C_TSO	0.79682 0.0179	0.02794 0.9476	0.94185 0.0005	0.97909 0.0001	0.99088 0.0001	0.99317 0.0001	0.98002 0.0001	0.98768 0.0001	0.99086 0.0001	0.98738 0.0001	0.99845 0.0001	-0.35678 0.3856	0.93768 0.0006
C_DH	1.00000 0.0000	0.32727 0.4288	0.90811 0.0018	0.88075 0.0039	0.83578 0.0098	0.83007 0.0108	0.86000 0.0062	0.84927 0.0076	0.84344 0.0085	0.84424 0.0084	0.80754 0.0153	-0.74834 0.0327	0.79828 0.0175
C_DA	0.32727 0.4288	1.00000 0.0000	0.29661 0.4756	0.16420 0.6976	0.15737 0.7098	0.13717 0.7460	0.19274 0.6475	0.16962 0.6880	0.14737 0.7277	0.17341 0.6813	0.05827 0.8810	-0.40007 0.3261	0.32710 0.4290
C_N	0.90811 0.0018	0.29661 0.4756	1.00000 0.0000	0.97154 0.0001	0.97262 0.0001	0.97082 0.0001	0.98295 0.0001	0.97936 0.0001	0.97583 0.0001	0.97882 0.0001	0.95152 0.0003	-0.52588 0.1807	0.96974 0.0001
C_NE	0.88075 0.0039	0.16420 0.6976	0.97154 0.0001	1.00000 0.0000	0.98685 0.0001	0.98930 0.0001	0.97958 0.0001	0.98685 0.0001	0.98640 0.0001	0.98466 0.0001	0.98217 0.0001	-0.43073 0.2867	0.94787 0.0003
M_N1	0.83578 0.0098	0.15737 0.7098	0.97262 0.0001	0.98685 0.0001	1.00000 0.0000	0.99894 0.0001	0.99621 0.0001	0.99933 0.0001	0.99946 0.0001	0.99948 0.0001	0.99472 0.0001	-0.42418 0.2949	0.97109 0.0001
M_N2	0.83007 0.0108	0.13717 0.7460	0.97082 0.0001	0.98930 0.0001	0.99894 0.0001	1.00000 0.0000	0.99261 0.0001	0.99796 0.0001	0.99852 0.0001	0.99734 0.0001	0.99625 0.0001	-0.39433 0.3337	0.96884 0.0001
M_BN1	0.86000 0.0062	0.19274 0.6475	0.98295 0.0001	0.97956 0.0001	0.99621 0.0001	0.99261 0.0001	1.00000 0.0000	0.99823 0.0001	0.99744 0.0001	0.99845 0.0001	0.99586 0.0001	-0.48723 0.2207	0.97192 0.0001

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	C_DH	C_DA	C_N	C_NE	M_N1	M_N2	M_BN1	M_BN2	M_V	M_D	M_E	M_L	M_VG
M_BN2	0.84927	0.16962	0.87936	0.98685	0.99933	0.99796	0.99823	1.00000	0.99971	0.99875	0.99227	-0.44488	0.97267
	0.0078	0.6880	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.2694	0.0001
M_V	0.84344	0.14737	0.87583	0.98640	0.99946	0.99852	0.99744	0.99971	1.00000	0.99948	0.99475	-0.43587	0.96863
	0.0085	0.7277	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.2804	0.0001
M_D	0.84424	0.17341	0.87682	0.98466	0.99948	0.99734	0.99845	0.99975	0.99946	1.00000	0.99203	-0.44730	0.97294
	0.0084	0.6913	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.2665	0.0001
M_E	0.80754	0.05827	0.95152	0.98217	0.99472	0.99625	0.98586	0.99227	0.99475	0.99203	1.00000	-0.37385	0.94755
	0.0153	0.8910	0.0003	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.3616	0.0003
M_L	-0.74834	-0.40007	-0.52588	-0.43073	-0.42418	-0.39433	-0.48723	-0.44488	-0.43587	-0.44730	-0.37385	1.00000	-0.39885
	0.0327	0.3261	0.1807	0.2867	0.2949	0.3337	0.2207	0.2694	0.2804	0.2665	0.3616	0.0000	0.3277
M_VG	0.79828	0.32710	0.96974	0.94787	0.97109	0.96884	0.97192	0.97267	0.96863	0.97294	0.94755	-0.39885	1.00000
	0.0176	0.4290	0.0001	0.0003	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0003	0.3277	0.0000
M_HK	0.55291	0.85389	0.67655	0.54524	0.57979	0.56337	0.61029	0.59022	0.57325	0.58407	0.49855	-0.42750	0.73163
	0.1552	0.0070	0.0654	0.1622	0.1319	0.1458	0.1081	0.1235	0.1374	0.1204	0.2086	0.2907	0.0391
M_TSD	0.66758	0.12320	0.80578	0.90358	0.87551	0.88558	0.83590	0.86365	0.86350	0.86247	0.87935	-0.12135	0.85114
	0.0705	0.7713	0.0158	0.0021	0.0044	0.0034	0.0097	0.0057	0.0057	0.0059	0.0040	0.7747	0.0074
M_DH	0.66638	0.03891	0.95180	0.98820	0.97914	0.98263	0.97170	0.97853	0.98120	0.97820	0.98600	-0.43386	0.91578
	0.0054	0.8271	0.0003	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.2828	0.0014
M_DA	0.63746	0.47498	0.65555	0.53210	0.49544	0.50678	0.52229	0.51858	0.50687	0.50473	0.44815	-0.32395	0.60867
	0.0891	0.2343	0.0776	0.1746	0.2119	0.1999	0.1842	0.1879	0.1999	0.2021	0.2654	0.4338	0.1093
M_N	0.85320	0.17743	0.88095	0.98481	0.99868	0.99656	0.99922	0.99980	0.99935	0.99971	0.99052	-0.45924	0.97280
	0.0071	0.6742	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.2523	0.0001
M_NE	0.82592	0.12467	0.86685	0.98706	0.99930	0.99856	0.99327	0.99798	0.99888	0.99784	0.99768	-0.40056	0.96518
	0.0115	0.7686	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.3254	0.0001

	M_HK	M_TSD	M_DH	M_DA	M_N	M_NE
A_N1	0.86358	0.84111	0.95214	0.53599	0.99389	0.98853
	0.0728	0.0088	0.0003	0.1709	0.0001	0.0001
A_N2	0.63943	0.87540	0.96265	0.52658	0.99550	0.99403
	0.0878	0.0044	0.0001	0.1780	0.0001	0.0001
A_BN1	0.82917	0.78796	0.95170	0.51586	0.99120	0.98274
	0.0847	0.0202	0.0003	0.1907	0.0001	0.0001
A_BN2	0.83716	0.82385	0.95838	0.53017	0.99610	0.98999
	0.0893	0.0118	0.0002	0.1765	0.0001	0.0001
A_V	0.81771	0.83778	0.96509	0.51509	0.99761	0.99402
	0.1027	0.0094	0.0001	0.1914	0.0001	0.0001

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	M_HK	M_TSO	M_DH	M_DA	M_N	M_NE
A_D	0.85653 0.0770	0.91488 0.0137	0.95145 0.0003	0.53269 0.1741	0.99354 0.0001	0.99591 0.0001
A_E	0.82248 0.1841	0.87833 0.0043	0.98324 0.0001	0.45815 0.2536	0.98323 0.0001	0.99680 0.0001
A_L	-0.42948 0.2760	0.08065 0.8496	-0.28838 0.4885	-0.38121 0.3515	-0.40032 0.3257	-0.35034 0.3949
A_VG	0.78427 0.0212	0.81756 0.0132	0.88507 0.0035	0.60586 0.1116	0.95758 0.0002	0.94682 0.0004
A_HK	0.95693 0.0002	0.59288 0.1214	0.56346 0.1458	0.63248 0.0924	0.69818 0.0541	0.66783 0.0703
A_TSO	0.81687 0.1896	0.87853 0.0044	0.96407 0.0001	0.45348 0.2591	0.98120 0.0001	0.98087 0.0001
A_DH	0.44150 0.2735	0.81533 0.0136	0.98623 0.0001	0.43930 0.2762	0.97952 0.0001	0.98352 0.0001
A_DA	-0.23557 0.5744	-0.37796 0.3559	-0.02963 0.9445	0.42857 0.2894	-0.11030 0.7949	-0.14384 0.7340
A_N	0.83032 0.0939	0.81488 0.0138	0.95821 0.0002	0.52302 0.1835	0.99542 0.0001	0.98899 0.0001
A_NE	0.84944 0.0814	0.85814 0.0064	0.95842 0.0002	0.53001 0.1767	0.99552 0.0001	0.99224 0.0001
C_N1	0.88787 0.1254	0.87853 0.0041	0.98316 0.0001	0.56142 0.1476	0.98963 0.0001	0.98699 0.0001
C_N2	0.46291 0.2481	0.83877 0.0005	0.95880 0.0002	0.50823 0.1984	0.92703 0.0009	0.93808 0.0006
C_BN1	0.87764 0.0848	0.78370 0.0214	0.94865 0.0003	0.64115 0.0867	0.98094 0.0001	0.96510 0.0001
C_BN2	0.87511 0.0862	0.81778 0.0131	0.95268 0.0003	0.66322 0.0730	0.97980 0.0001	0.96690 0.0001
C_V	0.82640 0.0966	0.86106 0.0060	0.96447 0.0001	0.60650 0.1109	0.98498 0.0001	0.97610 0.0001
C_D	0.88781 0.0594	0.80981 0.0148	0.95016 0.0003	0.62994 0.0941	0.98648 0.0001	0.97363 0.0001
C_E	0.88617 0.1523	0.86496 0.0055	0.98327 0.0001	0.50265 0.2043	0.99712 0.0001	0.99910 0.0001
C_L	-0.55172 0.1583	-0.19914 0.6537	-0.47841 0.2305	-0.60144 0.1147	-0.48841 0.2194	-0.44161 0.2733

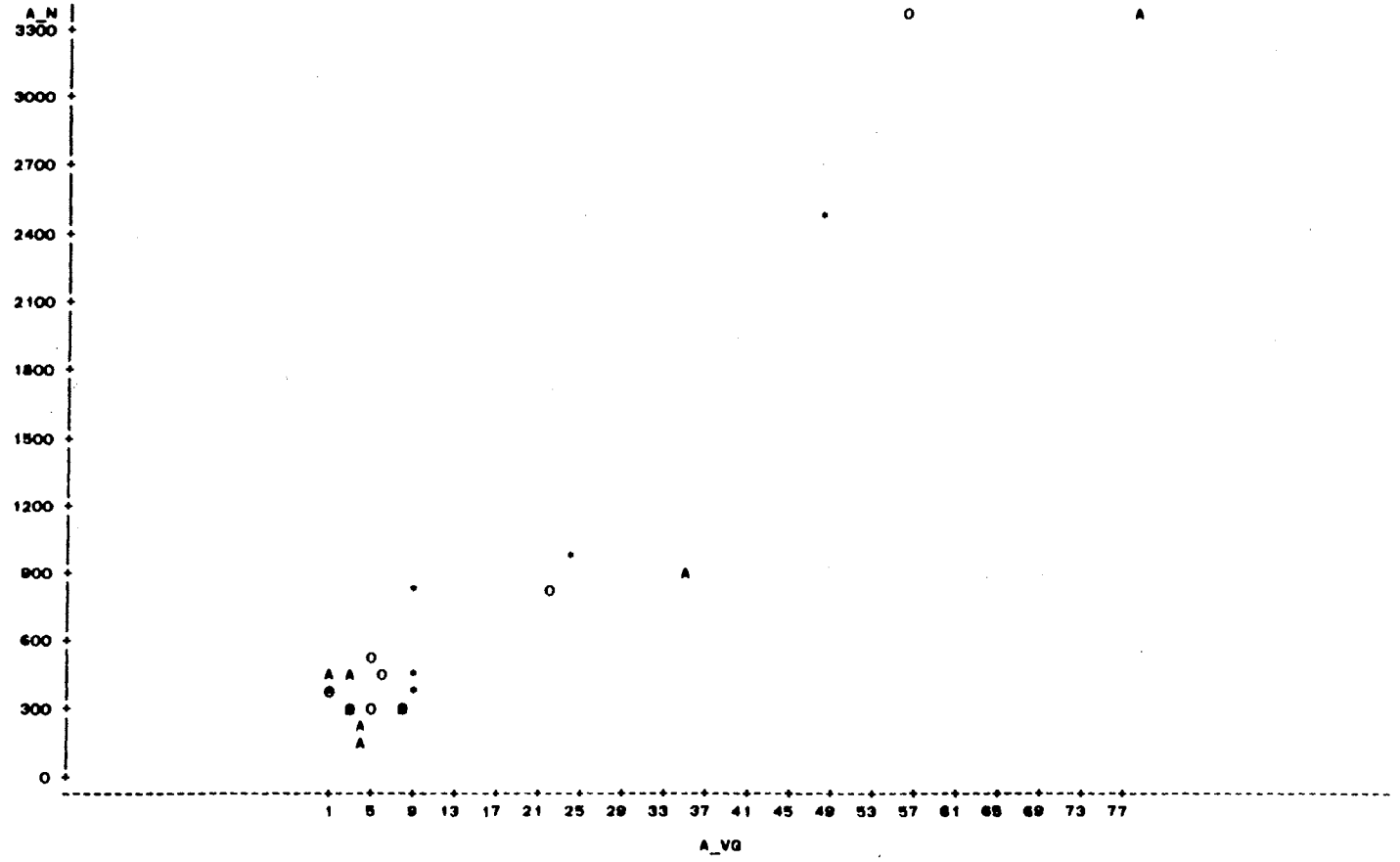
PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

	M_HK	M_TSO	M_DH	M_DA	M_N	M_NE
C_VG	0.77274 0.0246	0.84499 0.0083	0.88961 0.0031	0.68728 0.0596	0.94859 0.0003	0.93848 0.0006
C_HK	0.91034 0.0017	0.74357 0.0345	0.73598 0.0374	0.67723 0.0650	0.82958 0.0106	0.80604 0.0157
C_TSO	0.47104 0.2388	0.88192 0.0038	0.88615 0.0001	0.42857 0.2894	0.98550 0.0001	0.99497 0.0001
C_DH	0.85291 0.1552	0.86758 0.0705	0.86638 0.0054	0.63746 0.0891	0.85320 0.0071	0.82592 0.0115
C_DA	0.85388 0.0070	0.12320 0.7713	0.03891 0.9271	0.47498 0.2343	0.17743 0.6742	0.12467 0.7686
C_N	0.87656 0.0554	0.80678 0.0158	0.95190 0.0003	0.65555 0.0776	0.98095 0.0001	0.96695 0.0001
C_NE	0.84524 0.1622	0.90358 0.0021	0.98820 0.0001	0.53210 0.1746	0.98481 0.0001	0.98706 0.0001
M_N1	0.87878 0.1318	0.87551 0.0044	0.97914 0.0001	0.49544 0.2119	0.99868 0.0001	0.99930 0.0001
M_N2	0.86237 0.1458	0.88558 0.0034	0.98263 0.0001	0.50678 0.1999	0.99656 0.0001	0.99956 0.0001
M_SN1	0.81028 0.1061	0.83590 0.0087	0.97170 0.0001	0.52229 0.1842	0.99922 0.0001	0.99327 0.0001
M_SN2	0.89022 0.1235	0.86365 0.0057	0.97853 0.0001	0.51858 0.1879	0.99980 0.0001	0.99798 0.0001
M_V	0.87325 0.1374	0.86350 0.0057	0.98120 0.0001	0.50687 0.1999	0.99935 0.0001	0.99889 0.0001
M_D	0.89407 0.1204	0.86247 0.0059	0.97620 0.0001	0.50473 0.2021	0.99971 0.0001	0.99784 0.0001
M_E	0.49655 0.2066	0.87935 0.0040	0.98600 0.0001	0.44815 0.2654	0.99052 0.0001	0.99768 0.0001
M_L	-0.42750 0.2907	-0.12135 0.7747	-0.43386 0.2828	-0.32395 0.4338	-0.45924 0.2523	-0.40056 0.3254
M_VG	0.73163 0.0381	0.85114 0.0074	0.91578 0.0014	0.60867 0.1093	0.97280 0.0001	0.96519 0.0001
M_HK	1.00000 0.0000	0.44037 0.2748	0.44477 0.2695	0.64077 0.0869	0.59717 0.1180	0.55383 0.1644
M_TSO	0.44037 0.2748	1.00000 0.0000	0.86970 0.0050	0.37796 0.3559	0.85470 0.0069	0.88080 0.0039

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 8

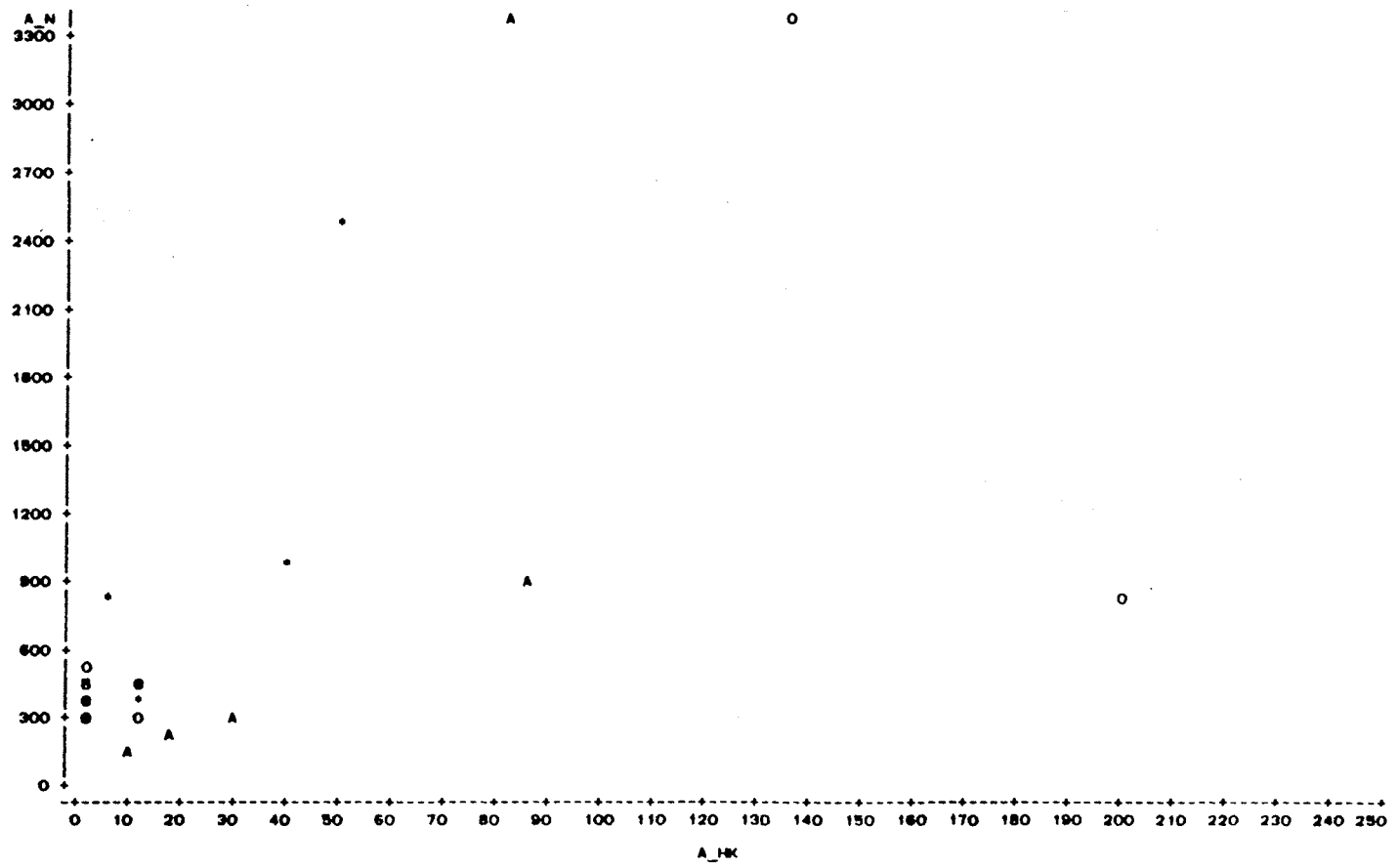
	M_HK	M_TSO	M_DH	M_DA	M_N	M_NE
M_DH	0.44477 0.2696	0.86970 0.0050	1.00000 0.0000	0.47626 0.2306	0.97663 0.0001	0.98270 0.0001
M_DA	0.64077 0.0869	0.37796 0.3559	0.47626 0.2306	1.00000 0.0000	0.52003 0.1865	0.48506 0.2231
M_N	0.59717 0.1180	0.85470 0.0069	0.97663 0.0001	0.52003 0.1865	1.00000 0.0000	0.99680 0.0001
M_NE	0.55383 0.1544	0.88080 0.0039	0.98270 0.0001	0.48506 0.2231	0.99680 0.0001	1.00000 0.0000

PLOT OF A_N*A_VQ LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF C_N*C_VQ SYMBOL USED IS *
PLOT OF M_N*M_VQ SYMBOL USED IS O



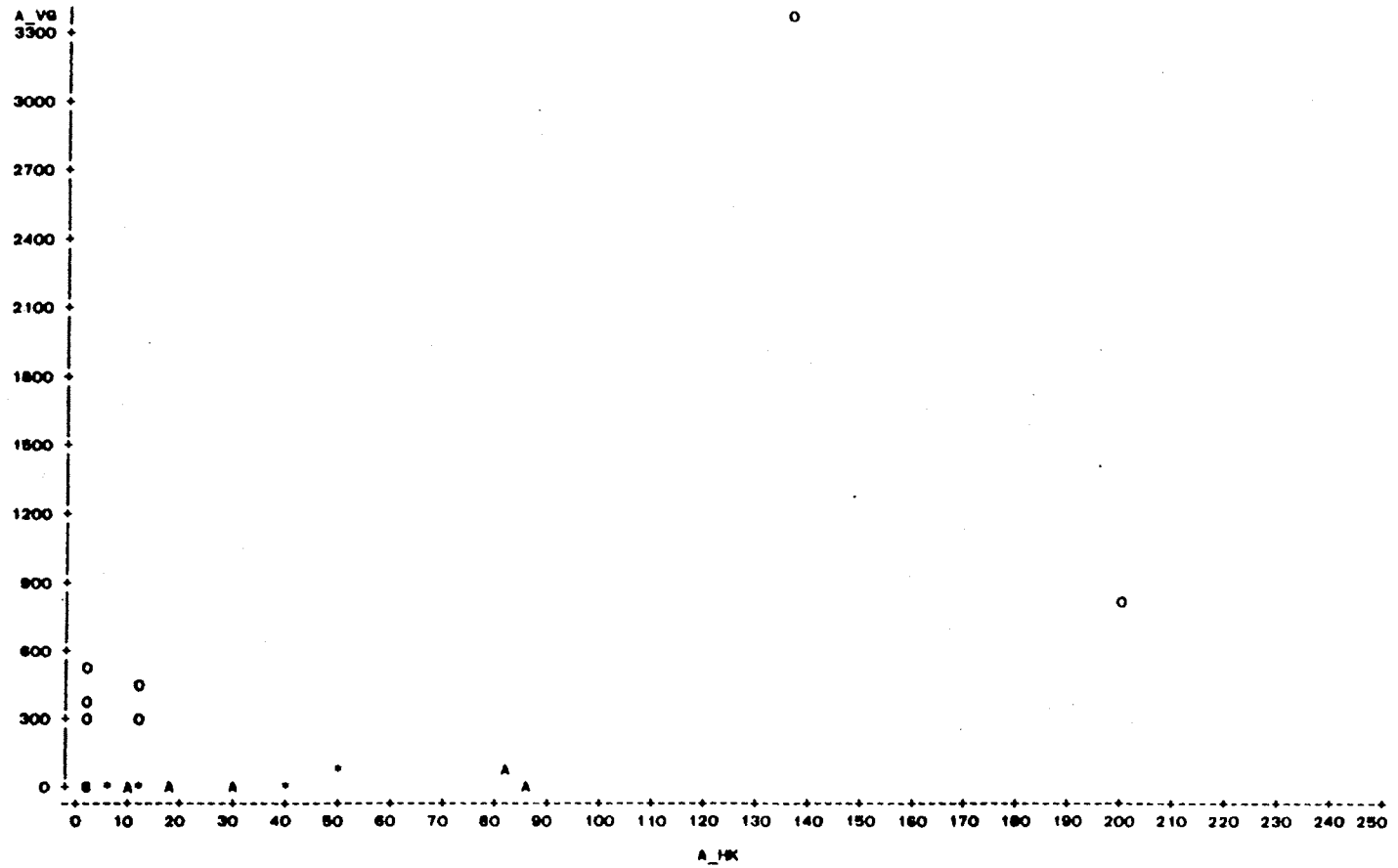
NOTE: 1 OBS HIDDEN

PLOT OF A_N*A_HK LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF C_N*C_HK SYMBOL USED IS *
PLOT OF M_N*M_HK SYMBOL USED IS O



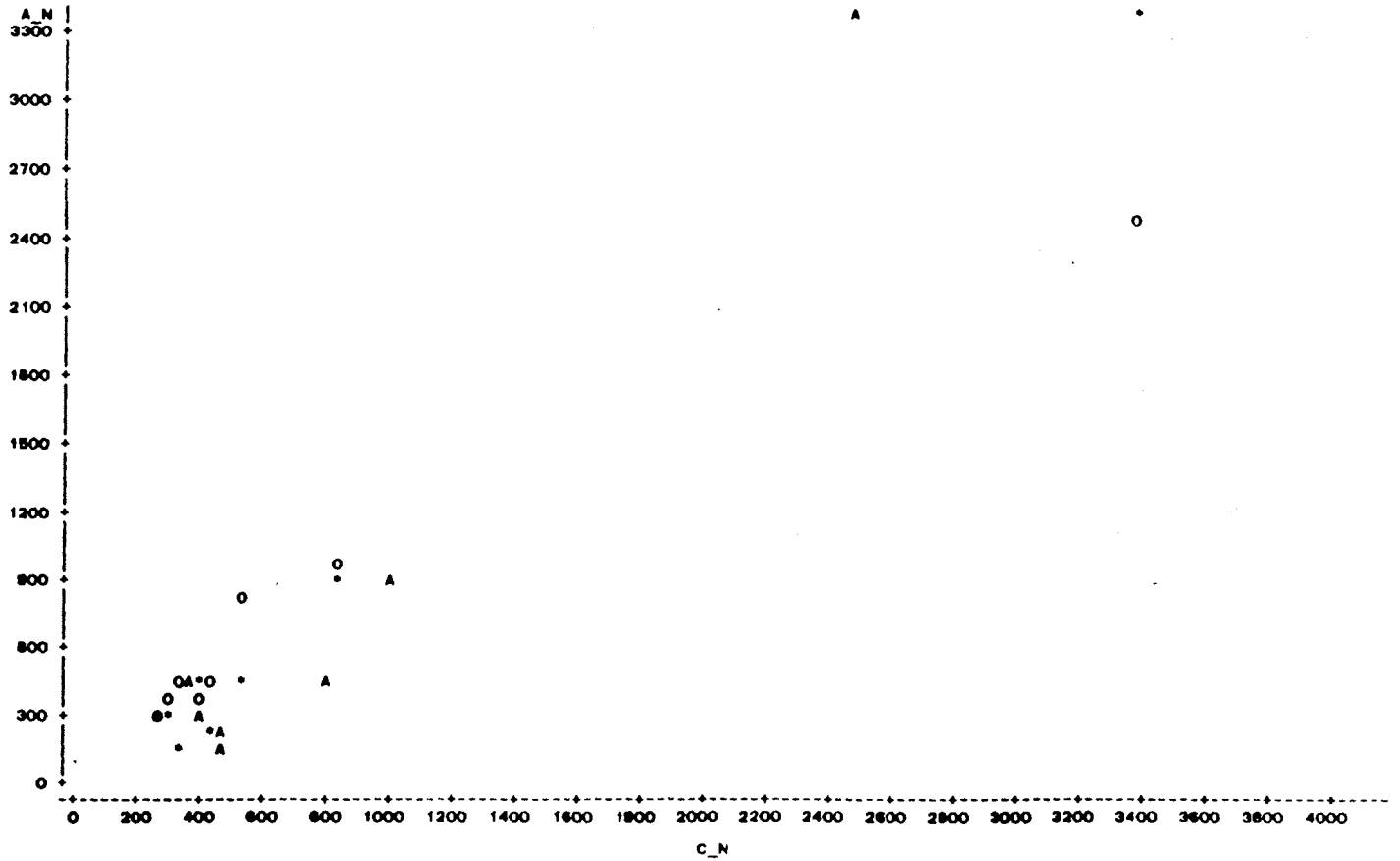
NOTE: 2 OBS HIDDEN

PLOT OF A_VG*A_HK LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF C_VG*C_HK SYMBOL USED IS *
PLOT OF M_N*M_HK SYMBOL USED IS O

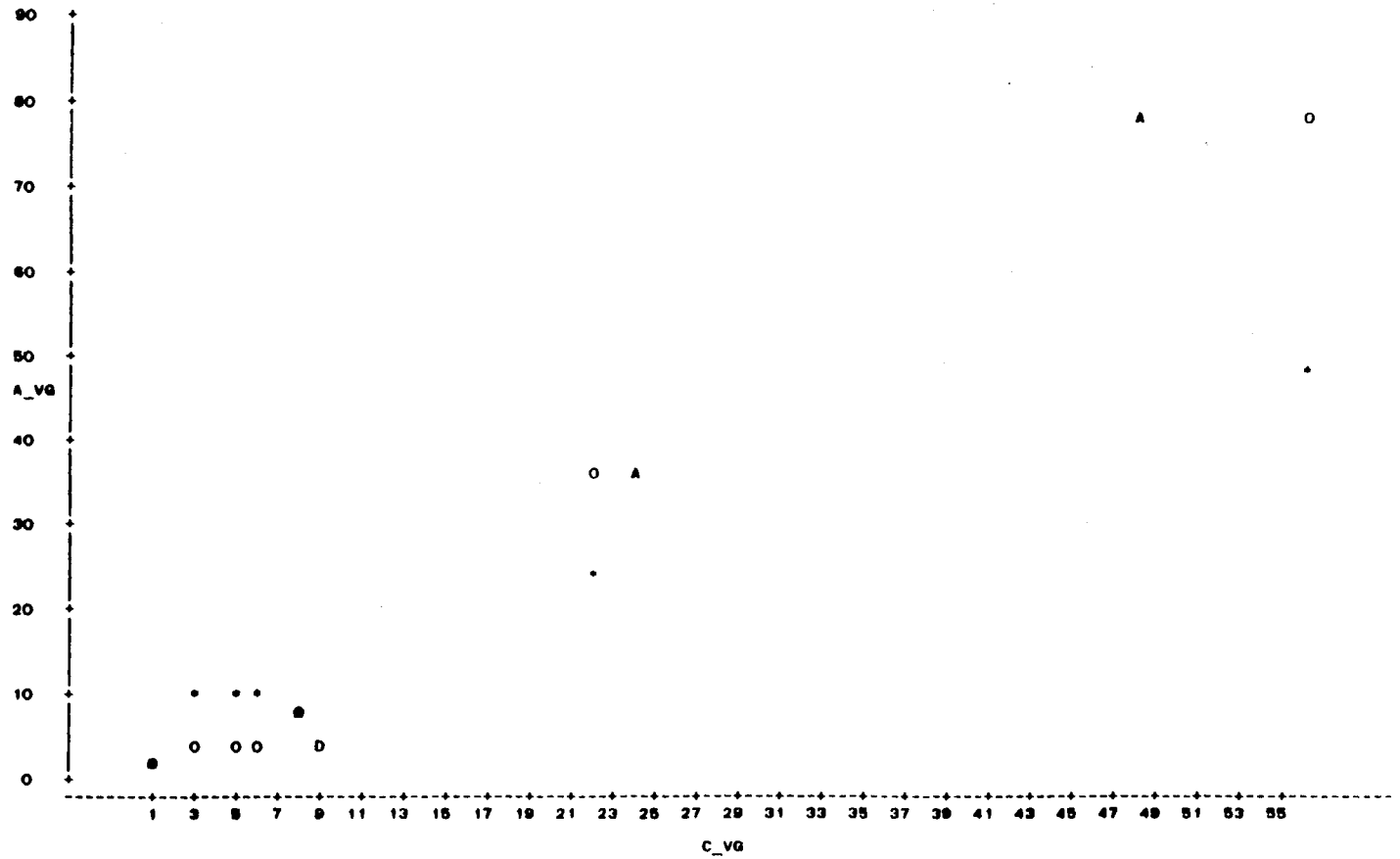


NOTE: 4 OBS HIDDEN

PLOT OF A_N*C_N LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF A_N*M_N SYMBOL USED IS *
PLOT OF C_N*M_N SYMBOL USED IS O

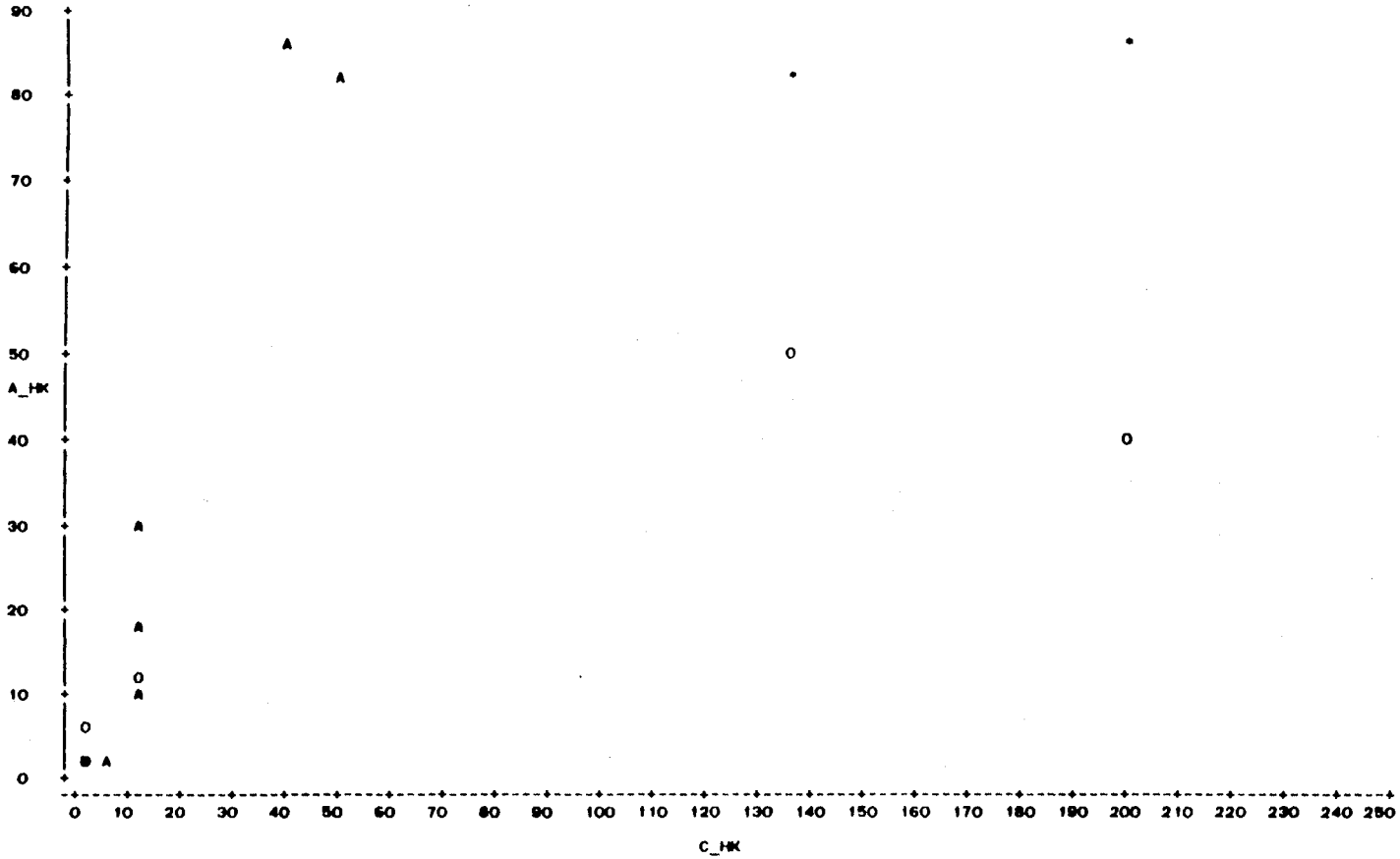


PLOT OF A_VG*C_VG LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF C_VG*M_VG SYMBOL USED IS *
PLOT OF A_VG*M_VG SYMBOL USED IS O



NOTE: 2 OBS HIDDEN

PLOT OF A_HK*C_HK LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF A_HK*M_HK SYMBOL USED IS *
PLOT OF C_HK*M_HK SYMBOL USED IS O



NOTE: 5 OBS HIDDEN

The following tables and plots are for the novice program analysis. The variables are represented as

l_m.

Where,

l could be

- A for Ada with the selected metric extrapolated with N; or
- Al for Ada with the selected metric extrapolated with Ne; or
- C for C++ with the selected metric extrapolated with N; or
- Cl for C++ with the selected metric extrapolated with Ne;

and

m is the selected metrics. (see nomenclature).

OBS	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VQ	A_N	A_NE
1	83	83	325	606	3756	311	386832	0.354	30.00	933	552
2	101	105	321	612	3706	294	281797	0.858	31.00	933	674
3	75	112	315	616	3550	207	176395	1.451	33.00	933	585
4	48	36	181	266	1756	171	181080	0.149	2.00	449	267
5	45	66	158	291	1632	92	52194	1.606	3.00	449	336
6	39	35	191	256	16989	142	142360	0.236	2.00	449	230
7	36	31	190	259	1758	151	198979	0.134	1.00	449	216
8	33	28	192	257	1698	155	189619	0.124	1.00	449	188
9	15	18	98	183	1082	74	112724	0.099	6.37	281	105
10	19	20	108	173	1063	79	94661	0.133	8.00	281	121
11	20	24	97	184	1069	73	80438	0.186	8.00	281	137
12	23	22	111	170	1089	92	112538	0.108	7.00	281	145
13	18	23	112	169	1043	66	67783	0.244	8.00	281	126
14	21	21	1101	170	1075	85	98978	0.120	9.00	281	133
15	70	70	130	196	1108	98	23286	2.515	5.00	326	377
16	28	37	118	208	1202	76	56886	0.835	3.00	326	195

OBS	A1_N1	A1_N2	A1_BN1	A1_BN2	A1_V	A1_D	A1_E	A1_L	A1_VG	A1_N	A1_NE
1	119	119	468	876	5406	434	556997	0.510	43	1343	795
2	118	123	379	722	4373	347	332575	0.777	37	1101	795
3	101	152	429	840	4826	282	239848	1.870	45	1269	795
4	28	22	109	162	1060	103	109293	0.080	1	271	161
5	21	33	76	140	782	44	25009	0.788	1	215	161
6	27	25	134	181	11815	99	99841	0.186	1	315	161
7	27	23	142	193	1310	113	148459	0.100	1	335	161
8	28	24	165	220	1455	133	162462	0.107	1	385	161
9	19	23	124	230	1365	93	142157	0.125	8	345	133
10	21	22	116	191	1169	87	104104	0.147	9	309	133
11	19	24	94	179	1037	71	78046	0.181	8	273	133
12	21	20	102	156	1011	84	103553	0.099	7	259	133
13	19	25	118	178	1098	70	71409	0.257	9	296	133
14	21	21	111	170	1074	85	88899	0.120	9	281	133
15	37	37	69	104	568	52	12346	1.330	2	173	200
16	28	38	122	213	1235	78	58457	0.550	3	335	200

OBS	C_N1	C_N2	C_BN1	C_BN2	C_V	C_D	C_E	C_L	C_VG	C_N	C_NE
1	163	96	406	603	4166	511	508402	0.251	25	1009	900
2	175	88	385	624	4472	624	889669	0.125	28	1009	1000
3	161	103	371	638	4315	503	589942	0.220	18	1009	954
4	158	112	367	642	4005	454	355627	0.510	46	1009	887
5	137	118	384	625	4022	365	308190	0.663	38	1009	843
6	27	27	157	199	1331	100	105119	0.215	1	356	162
7	48	58	78	193	1014	80	32214	1.004	10	271	321
8	35	29	99	172	1110	105	108490	0.129	8	271	219
9	48	60	106	165	1031	65	28058	1.380	5	271	339
10	28	19	94	177	1127	129	197669	0.037	7	271	164

OBS	C1_N1	C1_N2	C1_BN1	C1_BN2	C1_V	C1_D	C1_E	C1_L	C1_VG	C1_N	C1_NE
1	80	47	199	295	2038	250	248865	0.1230	12	494	440
2	77	39	170	275	1969	275	391851	0.0550	12	444	440
3	75	47	171	295	1991	232	272268	0.1020	8	466	440
4	78	56	182	319	1989	225	176579	0.2530	23	501	440
5	72	62	200	326	2100	190	160919	0.3460	20	527	440
6	30	30	177	226	1507	113	119047	0.2430	1	403	183
7	18	22	29	72	379	30	12036	0.3750	4	101	120
8	19	16	54	94	607	58	59344	0.0710	4	148	120
9	17	21	38	58	365	23	9939	0.4890	2	96	120
10	20	14	69	129	823	94	144442	0.0269	5	198	120

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
A_N1	16	42.0000000	26.4600328	672.000000	15.00000000	101.0000000
A_N2	16	45.8125000	31.4011014	733.000000	18.00000000	112.0000000
A_BN1	16	234.2500000	244.3575260	3748.000000	97.00000000	1101.0000000
A_BN2	16	289.0000000	165.5918676	4624.000000	168.00000000	618.0000000
A_V	16	2723.9375000	3929.6031282	43583.000000	1043.00000000	16988.0000000
A_D	16	135.3750000	77.4380387	2166.000000	66.00000000	311.0000000
A_E	16	141034.3750000	93884.0037889	2256550.000000	23288.00000000	386832.0000000
A_L	16	5.4075000000E-01	7.0402239555E-01	8.652000	8.9000000000E-02	2.5150000
A_VG	16	9.6356250	11.0083138	157.370000	1.0000000000E+00	33.0000000
A_N	16	461.3750000	244.9380534	7382.000000	281.00000000	933.0000000
A_NE	16	274.1875000	181.8539204	4387.000000	105.00000000	674.0000000
A1_N1	16	40.9375000	36.2720623	655.000000	18.00000000	119.0000000
A1_N2	16	45.6875000	43.3339263	731.000000	20.00000000	152.0000000
A1_BN1	16	172.5000000	128.5918606	2760.000000	69.00000000	468.0000000
A1_BN2	16	297.1875000	259.2186256	4755.000000	104.00000000	876.0000000
A1_V	16	2481.8250000	2946.3793120	39706.000000	588.00000000	11915.0000000
A1_D	16	135.9375000	113.8172031	2175.000000	44.00000000	434.0000000
A1_E	16	146467.3125000	134765.4890253	2343477.000000	12346.00000000	556997.0000000
A1_L	16	4.5612500000E-01	5.3516600023E-01	7.298000	9.0000000000E-02	1.9700000
A1_VG	16	11.5625000	15.3664949	185.000000	1.0000000000E+00	45.0000000
A1_N	16	469.0625000	387.3058448	7505.000000	173.00000000	1343.0000000
A1_NE	16	274.2500000	259.2617001	4388.000000	133.00000000	795.0000000
C_N1	10	98.0000000	65.1101631	980.000000	27.00000000	175.0000000
C_N2	10	71.0000000	37.3005809	710.000000	19.00000000	118.0000000
C_BN1	10	244.7000000	147.0722045	2447.000000	78.00000000	406.0000000
C_BN2	10	403.8000000	235.0559272	4038.000000	165.00000000	642.0000000
C_V	10	2659.3000000	1627.4128470	26593.000000	1014.00000000	4472.0000000
C_D	10	293.6000000	218.3596422	2936.000000	65.00000000	624.0000000
C_E	10	312338.0000000	280403.3947924	3123380.000000	28058.00000000	889469.0000000
C_L	10	4.5340000000E-01	4.4065890925E-01	4.534000	3.7000000000E-02	1.3800000
C_VG	10	18.6000000	15.1745401	186.000000	1.0000000000E+00	46.0000000
C_N	10	648.5000000	380.8444564	6485.000000	271.00000000	1008.0000000
C_NE	10	578.9000000	362.9118197	5789.000000	162.00000000	1000.0000000
C1_N1	10	46.6000000	29.5830282	466.000000	17.00000000	80.0000000
C1_N2	10	35.4000000	17.2059421	354.000000	14.00000000	62.0000000
C1_BN1	10	128.9000000	71.4958429	1289.000000	29.00000000	200.0000000
C1_BN2	10	206.9000000	108.7407212	2069.000000	58.00000000	326.0000000
C1_V	10	1376.8000000	745.2454927	13768.000000	365.00000000	2100.0000000
C1_D	10	148.0000000	96.0882828	1480.000000	23.00000000	275.0000000
C1_E	10	159508.0000000	120628.2595048	1595080.000000	9939.00000000	391851.0000000
C1_L	10	2.0839000000E-01	1.5726081886E-01	2.083900	2.6900000000E-02	4.6900000000E-01
C1_VG	10	9.1000000	7.5639027	91.000000	1.0000000000E+00	23.0000000
C1_N	10	337.8000000	179.1858450	3378.000000	96.00000000	527.0000000
C1_NE	10	288.3000000	163.0991859	2883.000000	120.00000000	440.0000000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / NUMBER OF OBSERVATIONS

	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VG	A_N	A_NE	A1_N1	A1_N2
A_N1	1.00000 0.0000 16	0.93524 0.0001 16	0.07446 0.7840 16	0.87466 0.0001 16	0.16812 0.4854 16	0.95579 0.0001 16	0.81408 0.0114 16	0.52494 0.0368 16	0.73149 0.0013 16	0.87946 0.0001 16	0.97830 0.0001 16	0.99238 0.0001 16	0.93364 0.0001 16
A_N2	0.93524 0.0001 16	1.00000 0.0000 16	0.06362 0.8148 16	0.89205 0.0001 16	0.12517 0.6442 16	0.73210 0.0013 16	0.45913 0.0736 16	0.63428 0.0083 16	0.78902 0.0003 16	0.87947 0.0001 16	0.87640 0.0001 16	0.86995 0.0001 16	0.90662 0.0001 16
A_BN1	0.07446 0.7840 16	0.06362 0.8148 16	1.00000 0.0000 16	0.12740 0.6382 16	-0.01167 0.9658 16	0.14124 0.6016 16	0.15530 0.5656 16	-0.10258 0.7054 16	0.23996 0.3707 16	0.12798 0.6367 16	0.08021 0.7397 16	0.15774 0.5596 16	0.13941 0.6066 16
A_BN2	0.87466 0.0001 16	0.89205 0.0001 16	0.12740 0.6382 16	1.00000 0.0000 16	0.20091 0.4556 16	0.91527 0.0001 16	0.77554 0.0004 16	0.23362 0.3839 16	0.88741 0.0001 16	0.99679 0.0001 16	0.93613 0.0001 16	0.96682 0.0001 16	0.95842 0.0001 16
A_V	0.16812 0.4854 16	0.12517 0.6442 16	-0.01167 0.9658 16	0.20091 0.4556 16	1.00000 0.0000 16	0.25999 0.3308 16	0.21199 0.4306 16	-0.07065 0.7949 16	0.03185 0.9068 16	0.23726 0.3763 16	0.16737 0.5355 16	0.14155 0.6010 16	0.10988 0.6854 16
A_D	0.95579 0.0001 16	0.73210 0.0013 16	0.14124 0.6016 16	0.91527 0.0001 16	0.25999 0.3308 16	1.00000 0.0000 16	0.92746 0.0001 16	0.04111 0.8799 16	0.74462 0.0009 16	0.93497 0.0001 16	0.95201 0.0001 16	0.91092 0.0001 16	0.79860 0.0002 16
A_E	0.81408 0.0114 16	0.45913 0.0736 16	0.15530 0.5656 16	0.77554 0.0004 16	0.21199 0.4306 16	0.92746 0.0001 16	1.00000 0.0000 16	-0.26147 0.2909 16	0.63125 0.0087 16	0.79994 0.0002 16	0.81426 0.0114 16	0.77405 0.0004 16	0.63015 0.0089 16
A_L	0.52494 0.0368 16	0.63428 0.0083 16	-0.10258 0.7054 16	0.23362 0.3839 16	-0.07065 0.7949 16	0.04111 0.8799 16	-0.26147 0.2909 16	1.00000 0.0000 16	0.17892 0.5073 16	0.21663 0.4203 16	0.50389 0.0466 16	0.23448 0.3821 16	0.32863 0.2139 16
A_VG	0.73149 0.0013 16	0.78902 0.0003 16	0.23996 0.3707 16	0.88741 0.0001 16	0.03185 0.9068 16	0.74462 0.0009 16	0.63125 0.0087 16	0.17892 0.5073 16	1.00000 0.0000 16	0.85878 0.0001 16	0.61341 0.0001 16	0.83083 0.0001 16	0.94915 0.0001 16
A_N	0.87946 0.0001 16	0.87947 0.0001 16	0.12798 0.6367 16	0.99679 0.0001 16	0.23726 0.3763 16	0.93497 0.0001 16	0.79994 0.0002 16	0.21663 0.4203 16	0.85878 0.0001 16	1.00000 0.0000 16	0.93196 0.0001 16	0.95910 0.0001 16	0.84054 0.0001 16
A_NE	0.97830 0.0001 16	0.97640 0.0001 16	0.08021 0.7397 16	0.93613 0.0001 16	0.16737 0.5355 16	0.95201 0.0001 16	0.81426 0.0114 16	0.50389 0.0466 16	0.61341 0.0001 16	0.93196 0.0001 16	1.00000 0.0000 16	0.92936 0.0001 16	0.91178 0.0001 16
A1_N1	0.99238 0.0001 16	0.86995 0.0001 16	0.15774 0.5596 16	0.96682 0.0001 16	0.14155 0.6010 16	0.91092 0.0001 16	0.77405 0.0004 16	0.23448 0.3821 16	0.83083 0.0001 16	0.95910 0.0001 16	0.92936 0.0001 16	1.00000 0.0000 16	0.95964 0.0001 16
A1_N2	0.93364 0.0001 16	0.90662 0.0001 16	0.13941 0.6066 16	0.95842 0.0001 16	0.10988 0.6854 16	0.79860 0.0002 16	0.63015 0.0089 16	0.32863 0.2139 16	0.94915 0.0001 16	0.84054 0.0001 16	0.91178 0.0001 16	0.95964 0.0001 16	1.00000 0.0000 16

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VG	A_N	A_NE	A1_N1	A1_N2
A1_BN1	0.75677	0.76821	0.17500	0.95337	0.16175	0.88251	0.81892	0.06764	0.92657	0.94798	0.82498	0.95990	0.94652
	0.0007	0.0005	0.5168	0.0001	0.5495	0.0001	0.0001	0.8034	0.0001	C.0001	0.0001	0.0001	0.0001
	16	16	16	16	16	16	16	16	16	16	16	16	16
A1_BN2	0.76012	0.79530	0.18391	0.95522	0.12106	0.85190	0.77194	0.11518	0.95232	0.94151	0.83814	0.96359	0.96611
	0.0006	0.0002	0.5441	0.0001	0.6552	0.0001	0.0005	0.6711	0.0001	0.0001	0.0001	0.0001	0.0001
	16	16	16	16	16	16	16	16	16	16	16	16	16
A1_V	0.37060	0.32351	0.04632	0.45443	0.95121	0.47835	0.42600	-0.06318	0.32088	0.48216	0.37664	0.41233	0.38318
	0.1576	0.2216	0.8647	0.0770	0.0001	0.0609	0.0999	0.8162	0.2256	0.0586	0.1504	0.1125	0.1429
	16	16	16	16	16	16	16	16	16	16	16	16	16
A1_D	0.77795	0.72084	0.18056	0.93104	0.15313	0.93602	0.89822	-0.00094	0.88971	0.93006	0.81586	0.96108	0.88660
	0.0004	0.0016	0.5034	0.0001	0.5713	0.0001	0.0001	0.9972	0.0001	0.0001	0.0001	0.0001	0.0001
	16	16	16	16	16	16	16	16	16	16	16	16	16
A1_E	0.63664	0.54309	0.17452	0.82194	0.12390	0.88740	0.94155	-0.16717	0.78531	0.82490	0.66330	0.86001	0.74959
	0.0080	0.0297	0.5180	0.0001	0.6475	0.0001	0.0001	0.5360	0.0003	0.0001	0.0051	0.0001	0.0008
	16	16	16	16	16	16	16	16	16	16	16	16	16
A1_L	0.66194	0.83804	-0.00480	0.58334	-0.01190	0.30686	-0.00002	0.83353	0.57930	0.55682	0.71614	0.56736	0.71907
	0.0052	0.0001	0.9859	0.0177	0.9651	0.2477	0.9999	0.0001	0.0187	0.0244	0.0018	0.0219	0.0017
	16	16	16	16	16	16	16	16	16	16	16	16	16
A1_VG	0.70455	0.76455	0.22091	0.89645	0.04151	0.75895	0.67347	0.13075	0.99290	0.87137	0.79212	0.93062	0.95099
	0.0023	0.0006	0.4110	0.0001	0.8787	0.0007	0.0042	0.6293	0.0001	0.0001	0.0003	0.0001	0.0001
	16	16	16	16	16	16	16	16	16	16	16	16	16
A1_N	0.76117	0.78925	0.18857	0.95638	0.13531	0.86401	0.78867	0.10019	0.94525	0.94556	0.83586	0.96418	0.96133
	0.0006	0.0003	0.5326	0.0001	0.6173	0.0001	0.0003	0.7120	0.0001	0.0001	0.0001	0.0001	0.0001
	16	16	16	16	16	16	16	16	16	16	16	16	16
A1_NE	0.86169	0.98250	0.15671	0.97513	0.12793	0.87379	0.73243	0.24879	0.95267	0.96237	0.92160	0.99186	0.96588
	0.0001	0.0001	0.5622	0.0001	0.6368	0.0001	0.0013	0.3568	0.0001	0.0001	0.0001	0.0001	0.0001
	16	16	16	16	16	16	16	16	16	16	16	16	16
C_N1	0.82898	0.82817	0.70957	0.78551	-0.21652	0.68351	0.47212	0.57095	0.68615	0.76716	0.84598	0.72356	0.72556
	0.0030	0.0030	0.0215	0.0071	0.5480	0.0293	0.1883	0.0847	0.0285	0.0096	0.0020	0.0180	0.0175
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_N2	0.57843	0.65106	0.45450	0.54534	-0.30467	0.39953	0.21715	0.63989	0.41813	0.52048	0.62021	0.43787	0.48806
	0.0798	0.0415	0.1870	0.1030	0.3920	0.2527	0.5468	0.0463	0.2292	0.1230	0.0558	0.2055	0.1524
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_BN1	0.78792	0.81534	0.87247	0.74512	-0.05116	0.62966	0.39651	0.63442	0.62969	0.72752	0.81742	0.66917	0.67681
	0.0057	0.0040	0.0331	0.0134	0.8884	0.0511	0.2566	0.0498	0.0511	0.0171	0.0039	0.0343	0.0316
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_BN2	0.77415	0.80624	0.64797	0.71990	-0.15481	0.58359	0.35242	0.65106	0.60202	0.70234	0.79895	0.63345	0.65826
	0.0088	0.0048	0.0428	0.0189	0.6684	0.0704	0.3178	0.0415	0.0655	0.0235	0.0056	0.0493	0.0385
	10	10	10	10	10	10	10	10	10	10	10	10	10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VG	A_N	A_NE	A1_N1	A1_N2
C_V	0.82366	0.84834	0.69826	0.77397	-0.12318	0.64493	0.39898	0.64852	0.66810	0.78561	0.84753	0.69845	0.71600
	0.0034	0.0018	0.0247	0.0086	0.7346	0.0441	0.2534	0.0425	0.0347	0.0115	0.0020	0.0247	0.0189
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_D	0.89760	0.87053	0.77798	0.84340	-0.13015	0.75621	0.52803	0.52856	0.76534	0.82682	0.90247	0.80428	0.76780
	0.0004	0.0010	0.0081	0.0022	0.7201	0.0110	0.1167	0.1154	0.0099	0.0030	0.0003	0.0050	0.0068
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_E	0.92662	0.89527	0.80615	0.87473	-0.07100	0.77537	0.52906	0.46329	0.84629	0.85906	0.93238	0.86737	0.84582
	0.0001	0.0005	0.0046	0.0009	0.8455	0.0084	0.1156	0.1570	0.0018	0.0015	0.0001	0.0011	0.0020
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_L	-0.44425	-0.37743	-0.47422	-0.40878	-0.28643	-0.43966	-0.29288	-0.11816	-0.36374	-0.43259	-0.41316	-0.42360	-0.38674
	0.1983	0.2823	0.1661	0.2406	0.4224	0.2036	0.4113	0.7451	0.2736	0.2118	0.2353	0.2225	0.2669
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_VG	0.46757	0.43322	0.28775	0.33085	-0.34082	0.33081	0.16266	0.42693	0.18283	0.31958	0.45333	0.25232	0.22506
	0.1730	0.2111	0.4201	0.3504	0.3356	0.3503	0.6535	0.2185	0.6132	0.3660	0.1882	0.4819	0.5318
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_N	0.78584	0.81185	0.65962	0.73206	-0.11530	0.60952	0.37064	0.64683	0.61474	0.71443	0.80878	0.64938	0.66764
	0.0070	0.0043	0.0380	0.0161	0.7511	0.0614	0.2917	0.0432	0.0586	0.0203	0.0046	0.0422	0.0349
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_NE	0.80840	0.83353	0.68611	0.77187	-0.24158	0.64374	0.42480	0.61554	0.67299	0.75030	0.83628	0.70058	0.71850
	0.0046	0.0027	0.0285	0.0089	0.5013	0.0446	0.2209	0.0582	0.0329	0.0124	0.0026	0.0240	0.0192
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_N1	0.80617	0.80676	0.68626	0.74690	-0.06111	0.65018	0.41841	0.59913	0.62795	0.73313	0.81800	0.67313	0.67817
	0.0048	0.0048	0.0284	0.0131	0.8668	0.0418	0.2276	0.0670	0.0519	0.0158	0.0038	0.0328	0.0318
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_N2	0.54680	0.61040	0.41980	0.47606	-0.01196	0.35020	0.12782	0.67086	0.30801	0.48169	0.57551	0.35024	0.39548
	0.1018	0.0609	0.2271	0.1643	0.9738	0.3212	0.7247	0.0337	0.3850	0.1762	0.0617	0.3211	0.2560
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_BN1	0.66618	0.66762	0.56218	0.58562	0.36580	0.50859	0.26358	0.56245	0.44707	0.58248	0.66858	0.50328	0.50486
	0.0364	0.0348	0.0907	0.0753	0.2866	0.1360	0.4618	0.0906	0.1952	0.0772	0.0346	0.1381	0.1367
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_BN2	0.68119	0.70459	0.56333	0.60033	0.18466	0.49561	0.23381	0.62486	0.46349	0.59299	0.69387	0.50527	0.52773
	0.0301	0.0229	0.0899	0.0665	0.6095	0.1452	0.5154	0.0535	0.1773	0.0708	0.0260	0.1363	0.1168
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_V	0.73843	0.78533	0.62183	0.66348	0.20500	0.55536	0.29030	0.62513	0.53733	0.65517	0.75058	0.57896	0.58395
	0.0147	0.0115	0.0549	0.0365	0.5699	0.0956	0.4158	0.0533	0.1092	0.0398	0.0124	0.0785	0.0702
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_D	0.86424	0.83300	0.74556	0.79512	0.04139	0.72112	0.47714	0.52788	0.70896	0.78582	0.86417	0.75271	0.73288
	0.0013	0.0028	0.0133	0.0059	0.9096	0.0188	0.1632	0.1189	0.0215	0.0070	0.0013	0.0120	0.0157
	10	10	10	10	10	10	10	10	10	10	10	10	10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / NUMBER OF OBSERVATIONS

	A_N1	A_N2	A_BN1	A_BN2	A_V	A_D	A_E	A_L	A_VG	A_N	A_NE	A1_N1	A1_N2
C1_E	0.88788 0.0004 10	0.96531 0.0012 10	0.77626 0.0083 10	0.83863 0.0024 10	0.08426 0.8600 10	0.74231 0.0139 10	0.48524 0.1551 10	0.47455 0.1656 10	0.82096 0.0036 10	0.82498 0.0033 10	0.90044 0.0004 10	0.83442 0.0027 10	0.91222 0.0043 10
C1_L	-0.47003 0.1704 10	-0.40193 0.2496 10	-0.50863 0.1322 10	-0.47388 0.1665 10	-0.03038 0.9336 10	-0.49721 0.1437 10	-0.39169 0.2630 10	-0.05044 0.8899 10	-0.50911 0.1328 10	-0.48880 0.1517 10	-0.44704 0.1952 10	-0.52218 0.1215 10	-0.48328 0.1570 10
C1_VG	0.38006 0.2787 10	0.35875 0.3072 10	0.19686 0.5856 10	0.24270 0.4993 10	-0.32836 0.3527 10	0.23922 0.5056 10	0.07757 0.8313 10	0.42272 0.2236 10	0.10345 0.7761 10	0.22992 0.5226 10	0.36887 0.2841 10	0.18227 0.6542 10	0.14151 0.6966 10
C1_N	0.67804 0.0312 10	0.69341 0.0262 10	0.56504 0.0886 10	0.59696 0.0684 10	0.25778 0.4721 10	0.50104 0.1402 10	0.24567 0.4938 10	0.60460 0.0641 10	0.45845 0.1827 10	0.58121 0.0719 10	0.69690 0.0282 10	0.50609 0.1356 10	0.52069 0.1228 10
C1_NE	0.78830 0.0066 10	0.81266 0.0043 10	0.66429 0.0362 10	0.73199 0.0161 10	-0.06730 0.8534 10	0.61162 0.0602 10	0.36798 0.2955 10	0.64685 0.0432 10	0.60878 0.0612 10	0.71587 0.0199 10	0.81037 0.0045 10	0.64759 0.0429 10	0.66523 0.0358 10

	A1_BN1	A1_BN2	A1_V	A1_D	A1_E	A1_L	A1_VG	A1_N	A1_NE	C_N1	C_N2	C_BN1	C_BN2
A_N1	0.75677 0.0007 16	0.76012 0.0006 16	0.37060 0.1576 16	0.77785 0.0004 16	0.63664 0.0080 16	0.66194 0.0052 16	0.70455 0.0023 16	0.76117 0.0006 16	0.96169 0.0001 16	0.92898 0.0030 10	0.57843 0.0799 10	0.79792 0.0057 10	0.77415 0.0086 10
A_N2	0.76821 0.0005 16	0.79530 0.0002 16	0.32351 0.2216 16	0.72084 0.0016 16	0.54309 0.0297 16	0.83804 0.0001 16	0.76455 0.0006 16	0.78825 0.0003 16	0.88250 0.0001 16	0.82917 0.0030 10	0.65105 0.0415 10	0.81534 0.0040 10	0.80524 0.0048 10
A_BN1	0.17500 0.5168 16	0.16391 0.5441 16	0.04632 0.8647 16	0.15056 0.5034 16	0.17452 0.5180 16	-0.00480 0.9658 16	0.22091 0.4110 16	0.16857 0.5328 16	0.15671 0.5622 16	0.70967 0.0215 10	0.45460 0.1870 10	0.67247 0.0331 10	0.64787 0.0428 10
A_BN2	0.85337 0.0001 16	0.85522 0.0001 16	0.45443 0.0770 16	0.83104 0.0001 16	0.82184 0.0001 16	0.58334 0.0177 16	0.89645 0.0001 16	0.95639 0.0001 16	0.97513 0.0001 16	0.78551 0.0071 10	0.54534 0.1030 10	0.74512 0.0134 10	0.71990 0.0188 10
A_V	0.16175 0.5485 16	0.12106 0.6552 16	0.85121 0.0001 16	0.15313 0.5713 16	0.12390 0.6475 16	-0.01190 0.9651 16	0.04151 0.8787 16	0.13531 0.6173 16	0.12793 0.6368 16	-0.21652 0.5480 10	-0.30467 0.3920 10	-0.05116 0.8884 10	-0.15481 0.6684 10
A_D	0.88251 0.0001 16	0.85190 0.0001 16	0.47835 0.0608 16	0.83602 0.0001 16	0.88740 0.0001 16	0.30686 0.2477 16	0.75895 0.0007 16	0.86401 0.0001 16	0.87378 0.0001 16	0.68351 0.0293 10	0.39953 0.2527 10	0.62966 0.0511 10	0.58359 0.0704 10
A_E	0.81662 0.0001 16	0.77184 0.0005 16	0.42600 0.0999 16	0.89822 0.0001 16	0.94155 0.0001 16	-0.00002 0.9999 16	0.87347 0.0042 16	0.78867 0.0003 16	0.73243 0.0013 16	0.47212 0.1683 10	0.21715 0.5468 10	0.39651 0.2568 10	0.36242 0.3178 10
A_L	0.06764 0.8034 16	0.11516 0.6711 16	-0.06319 0.8182 16	-0.00094 0.9972 16	-0.16717 0.5360 16	0.83393 0.0001 16	0.13075 0.6293 16	0.10019 0.7120 16	0.24679 0.3568 16	0.87095 0.0047 10	0.63988 0.0463 10	0.63442 0.0488 10	0.65106 0.0415 10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	A1_BN1	A1_BN2	A1_V	A1_D	A1_E	A1_L	A1_VG	A1_N	A1_NE	C_N1	C_N2	C_BN1	C_BN2
A_VG	0.92657 0.0001 16	0.95232 0.0001 16	0.32086 0.2256 16	0.88971 0.0001 16	0.78531 0.0003 16	0.57930 0.0187 16	0.99290 0.0001 16	0.94525 0.0001 16	0.95267 0.0001 16	0.68615 0.0285 10	0.41813 0.2292 10	0.62969 0.0511 10	0.60202 0.0655 10
A_N	0.94798 0.0001 16	0.94151 0.0001 16	0.48216 0.0586 16	0.93006 0.0001 16	0.82490 0.0001 16	0.55882 0.0244 16	0.87137 0.0001 16	0.84558 0.0001 16	0.96237 0.0001 16	0.76716 0.0096 10	0.52048 0.1230 10	0.72752 0.0171 10	0.70234 0.0235 10
A_NE	0.82498 0.0001 16	0.83814 0.0001 16	0.37664 0.1504 16	0.81586 0.0001 16	0.66330 0.0051 16	0.71614 0.0018 16	0.79212 0.0003 16	0.83586 0.0001 16	0.82160 0.0001 16	0.84598 0.0020 10	0.62021 0.0558 10	0.81742 0.0039 10	0.79895 0.0056 10
A1_N1	0.95990 0.0001 16	0.96359 0.0001 16	0.41233 0.1125 16	0.96108 0.0001 16	0.86001 0.0001 16	0.56736 0.0219 16	0.93062 0.0001 16	0.96418 0.0001 16	0.99186 0.0001 16	0.72356 0.0180 10	0.43797 0.2055 10	0.66917 0.0343 10	0.63345 0.0493 10
A1_N2	0.94652 0.0001 16	0.96611 0.0001 16	0.38319 0.1429 16	0.88680 0.0001 16	0.74959 0.0008 16	0.71907 0.0017 16	0.95099 0.0001 16	0.96133 0.0001 16	0.98588 0.0001 16	0.72556 0.0175 10	0.48806 0.1524 10	0.67681 0.0316 10	0.65826 0.0385 10
A1_BN1	1.00000 0.0000 16	0.99363 0.0001 16	0.44955 0.0806 16	0.97429 0.0001 16	0.90652 0.0001 16	0.48845 0.0549 16	0.95312 0.0001 16	0.99727 0.0001 16	0.97014 0.0001 16	0.62683 0.0524 10	0.35508 0.3140 10	0.57017 0.0853 10	0.53339 0.1123 10
A1_BN2	0.99363 0.0001 16	1.00000 0.0000 16	0.41392 0.1110 16	0.96130 0.0001 16	0.88302 0.0001 16	0.53588 0.0324 16	0.97341 0.0001 16	0.99922 0.0001 16	0.98064 0.0001 16	0.66781 0.0348 10	0.41102 0.2380 10	0.61211 0.0600 10	0.57751 0.0804 10
A1_V	0.44955 0.0806 16	0.41392 0.1110 16	1.00000 0.0000 16	0.43198 0.0947 16	0.39117 0.1341 16	0.12752 0.6379 16	0.33893 0.1991 16	0.42672 0.0993 16	0.40545 0.1192 16	-0.03276 0.9284 10	-0.20021 0.5792 10	0.10700 0.7686 10	-0.00662 0.9855 10
A1_D	0.87429 0.0001 16	0.96130 0.0001 16	0.43198 0.0947 16	1.00000 0.0000 16	0.96388 0.0001 16	0.35576 0.1763 16	0.91214 0.0001 16	0.96706 0.0001 16	0.94566 0.0001 16	0.63186 0.0500 10	0.34147 0.3342 10	0.57401 0.0627 10	0.52483 0.1193 10
A1_E	0.90652 0.0001 16	0.88302 0.0001 16	0.39117 0.1341 16	0.96388 0.0001 16	1.00000 0.0000 16	0.14806 0.5842 16	0.82587 0.0001 16	0.89160 0.0001 16	0.83312 0.0001 16	0.50446 0.1370 10	0.24647 0.4888 10	0.44882 0.1932 10	0.38535 0.2715 10
A1_L	0.48845 0.0549 16	0.53588 0.0324 16	0.12752 0.6379 16	0.35576 0.1763 16	0.14806 0.5842 16	1.00000 0.0000 16	0.55405 0.0260 16	0.52156 0.0383 16	0.61774 0.0108 16	0.61138 0.0604 10	0.53615 0.1101 10	0.60132 0.0659 10	0.62389 0.0539 10
A1_VG	0.95312 0.0001 16	0.97341 0.0001 16	0.33893 0.1991 16	0.91214 0.0001 16	0.82587 0.0001 16	0.55405 0.0260 16	1.00000 0.0000 16	0.96801 0.0001 16	0.95532 0.0001 16	0.66313 0.0366 10	0.40507 0.2455 10	0.60621 0.0632 10	0.57651 0.0811 10
A1_N	0.98727 0.0001 16	0.99922 0.0001 16	0.42672 0.0993 16	0.96706 0.0001 16	0.89160 0.0001 16	0.52156 0.0383 16	0.96801 0.0001 16	1.00000 0.0000 16	0.97891 0.0001 16	0.65592 0.0385 10	0.39300 0.2612 10	0.60016 0.0666 10	0.56506 0.0887 10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	A1_BN1	A1_BN2	A1_V	A1_O	A1_E	A1_L	A1_VG	A1_N	A1_NE	C_N1	C_N2	C_BN1	C_BN2
A1_NE	0.87014 0.0001 16	0.98064 0.0001 16	0.40545 0.1192 16	0.94566 0.0001 16	0.83312 0.0001 16	0.61774 0.0108 16	0.95532 0.0001 16	0.97891 0.0001 16	1.00000 0.0000 16	0.73322 0.0158 10	0.46729 0.1733 10	0.68020 0.0304 10	0.65108 0.0414 10
C_N1	0.62683 0.0524 10	0.66781 0.0348 10	-0.03276 0.9284 10	0.63186 0.0500 10	0.50446 0.1370 10	0.61138 0.0604 10	0.66313 0.0366 10	0.65582 0.0395 10	0.73322 0.0158 10	1.00000 0.0000 10	0.90828 0.0003 10	0.96585 0.0001 10	0.98066 0.0001 10
C_N2	0.35506 0.3140 10	0.41102 0.2380 10	-0.20021 0.5782 10	0.34147 0.3342 10	0.24847 0.4886 10	0.53615 0.1101 10	0.40507 0.2455 10	0.39300 0.2612 10	0.46729 0.1733 10	0.90828 0.0003 10	1.00000 0.0000 10	0.88162 0.0007 10	0.91608 0.0002 10
C_BN1	0.57017 0.0853 10	0.61211 0.0600 10	0.10700 0.7686 10	0.57401 0.0827 10	0.44882 0.1932 10	0.60132 0.0658 10	0.60521 0.0632 10	0.60016 0.0666 10	0.68020 0.0304 10	0.96585 0.0001 10	0.88162 0.0007 10	1.00000 0.0000 10	0.98583 0.0001 10
C_BN2	0.53339 0.1123 10	0.57751 0.0804 10	-0.00662 0.9855 10	0.52483 0.1183 10	0.38535 0.2715 10	0.62389 0.0539 10	0.57651 0.0811 10	0.56506 0.0887 10	0.65108 0.0414 10	0.98066 0.0001 10	0.91608 0.0002 10	0.98583 0.0001 10	1.00000 0.0000 10
C_V	0.59764 0.0691 10	0.64046 0.0461 10	0.04525 0.9012 10	0.59073 0.0721 10	0.44574 0.1967 10	0.64810 0.0427 10	0.64033 0.0461 10	0.62643 0.0517 10	0.71245 0.0205 10	0.98418 0.0001 10	0.88441 0.0007 10	0.99072 0.0001 10	0.99436 0.0001 10
C_O	0.69937 0.0244 10	0.73227 0.0160 10	0.07183 0.8437 10	0.70952 0.0216 10	0.56166 0.0911 10	0.62081 0.0556 10	0.73362 0.0157 10	0.72373 0.0180 10	0.80625 0.0048 10	0.97245 0.0001 10	0.79088 0.0064 10	0.94802 0.0001 10	0.95124 0.0001 10
C_E	0.75844 0.0113 10	0.76505 0.0071 10	0.14838 0.6804 10	0.75864 0.0110 10	0.58080 0.0783 10	0.64131 0.0457 10	0.80139 0.0053 10	0.77603 0.0080 10	0.86813 0.0012 10	0.95766 0.0015 10	0.69055 0.0723 10	0.91821 0.0036 10	0.91462 0.0041 10
C_L	-0.40449 0.2463 10	-0.36908 0.2838 10	-0.37994 0.2788 10	-0.39636 0.2568 10	-0.26441 0.4258 10	-0.26220 0.4643 10	-0.36508 0.2996 10	-0.38547 0.2713 10	-0.40629 0.2453 10	-0.18947 0.6001 10	0.15184 0.6752 10	-0.27682 0.4388 10	-0.24278 0.4891 10
C_VG	0.11698 0.7476 10	0.15654 0.6658 10	-0.31856 0.3697 10	0.17492 0.6289 10	0.10173 0.7796 10	0.22785 0.5267 10	0.15177 0.6755 10	0.14518 0.6890 10	0.24404 0.4968 10	0.82567 0.0033 10	0.86786 0.0011 10	0.82072 0.0036 10	0.86522 0.0012 10
C_N	0.54838 0.1000 10	0.58282 0.0709 10	0.03723 0.9167 10	0.54559 0.1028 10	0.41116 0.2376 10	0.61728 0.0573 10	0.58993 0.0726 10	0.58052 0.0785 10	0.66452 0.0361 10	0.97825 0.0001 10	0.90586 0.0003 10	0.99463 0.0001 10	0.99790 0.0001 10
C_NE	0.60456 0.0641 10	0.65067 0.0418 10	-0.06331 0.8621 10	0.59924 0.0671 10	0.46630 0.1743 10	0.63636 0.0470 10	0.64888 0.0425 10	0.63685 0.0477 10	0.71896 0.0189 10	0.99651 0.0001 10	0.92965 0.0001 10	0.95866 0.0001 10	0.97607 0.0001 10
C1_N1	0.67422 0.0325 10	0.61192 0.0901 10	0.09656 0.7907 10	0.57921 0.0783 10	0.45081 0.1913 10	0.58079 0.0721 10	0.60505 0.0638 10	0.60168 0.0657 10	0.69187 0.0299 10	0.97286 0.0001 10	0.87724 0.0009 10	0.99539 0.0001 10	0.99107 0.0001 10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	A1_BN1	A1_BN2	A1_V	A1_D	A1_E	A1_L	A1_VQ	A1_N	A1_NE	C_N1	C_N2	C_BN1	C_BN2
C1_N2	0.28594	0.30339	0.04136	0.24586	0.14331	0.48596	0.29142	0.28924	0.37445	0.84592	0.92017	0.91427	0.91570
	0.4754	0.3941	0.9097	0.4935	0.6929	0.1544	0.4139	0.4176	0.2864	0.0020	0.0002	0.0002	0.0002
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_BN1	0.40932	0.43600	0.45218	0.41278	0.29439	0.47153	0.42322	0.42873	0.50789	0.74874	0.65662	0.88520	0.82719
	0.2401	0.2078	0.1895	0.2358	0.4090	0.1888	0.2230	0.2152	0.1339	0.0127	0.0392	0.0007	0.0031
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_BN2	0.40466	0.43913	0.27478	0.39571	0.25663	0.53614	0.43614	0.43059	0.51876	0.63632	0.75927	0.93686	0.81351
	0.2461	0.2042	0.4423	0.2577	0.4742	0.1101	0.2077	0.2142	0.1244	0.0026	0.0109	0.0001	0.0002
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_V	0.47787	0.51125	0.31858	0.47068	0.32666	0.56584	0.50825	0.50303	0.58995	0.84887	0.73884	0.94663	0.91407
	0.1624	0.1310	0.3696	0.1698	0.3569	0.0682	0.1336	0.1383	0.0726	0.0019	0.0146	0.0001	0.0002
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_D	0.64642	0.67560	0.21820	0.65806	0.51093	0.58492	0.67794	0.66881	0.75142	0.92448	0.73068	0.95529	0.93779
	0.0434	0.0320	0.5448	0.0386	0.1312	0.0757	0.0312	0.0345	0.0122	0.0001	0.0164	0.0001	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_E	0.72359	0.74998	0.26951	0.72501	0.54585	0.61781	0.77298	0.74415	0.83092	0.80580	0.51578	0.80947	0.78937
	0.0180	0.0125	0.4514	0.0177	0.1026	0.0570	0.0087	0.0136	0.0029	0.0049	0.1270	0.0045	0.0066
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_L	-0.51943	-0.48710	-0.18050	-0.51123	-0.40982	-0.29361	-0.49502	-0.50190	-0.50458	-0.21138	0.16256	-0.22323	-0.21296
	0.1239	0.1533	0.6178	0.1310	0.2395	0.4103	0.1458	0.1394	0.1369	0.5577	0.6536	0.5353	0.5547
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_VQ	0.03086	0.07219	-0.33239	0.08834	0.02655	0.18106	0.07387	0.06028	0.15590	0.76596	0.82585	0.78510	0.82632
	0.9326	0.8429	0.3480	0.8082	0.9420	0.6167	0.8393	0.8686	0.6871	0.0098	0.0032	0.0071	0.0032
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_N	0.40767	0.43932	0.34657	0.40346	0.27199	0.51355	0.43241	0.43161	0.51826	0.80800	0.72335	0.92184	0.88453
	0.2422	0.2040	0.3266	0.2476	0.4471	0.1289	0.2120	0.2130	0.1286	0.0049	0.0181	0.0001	0.0007
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_NE	0.54663	0.58887	0.08143	0.54233	0.40558	0.61867	0.58442	0.57702	0.66222	0.97104	0.89617	0.99643	0.99484
	0.1020	0.0733	0.8231	0.1053	0.2449	0.0581	0.0760	0.0807	0.0370	0.0001	0.0004	0.0001	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10

	C_V	C_D	C_E	C_L	C_VQ	C_N	C_NE	C1_N1	C1_N2	C1_BN1	C1_BN2	C1_V	C1_D
A_N1	0.82368	0.89760	0.92662	-0.44425	0.46757	0.78594	0.80940	0.80517	0.54680	0.66618	0.68119	0.73643	0.86424
	0.0034	0.0004	0.0001	0.1963	0.1730	0.0070	0.0046	0.0049	0.1019	0.0354	0.0301	0.0147	0.0013
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_N2	0.84834	0.87053	0.99527	-0.37743	0.43322	0.81165	0.83953	0.80676	0.61040	0.86762	0.70459	0.75533	0.83300
	0.0019	0.0010	0.0005	0.2823	0.2111	0.0043	0.0027	0.0048	0.0609	0.0349	0.0229	0.0115	0.0028
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_BN1	0.69626	0.77796	0.80515	-0.47422	0.28775	0.85962	0.88811	0.88628	0.41990	0.56216	0.56333	0.62193	0.74556
	0.0247	0.0081	0.0049	0.1661	0.4201	0.0060	0.0285	0.0284	0.2271	0.0807	0.0899	0.0549	0.0133
	10	10	10	10	10	10	10	10	10	10	10	10	10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	C_V	C_D	C_E	C_L	C_VG	C_N	C_NE	C1_N1	C1_N2	C1_BN1	C1_BN2	C1_V	C1_D
A_BN2	0.77397	0.84340	0.87473	-0.40878	0.33085	0.73208	0.77187	0.74690	0.47606	0.58562	0.60033	0.66349	0.79612
	0.0086	0.0022	0.0009	0.2408	0.3504	0.0161	0.0089	0.0131	0.1643	0.0753	0.0665	0.0365	0.0059
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_V	-0.12318	-0.13015	-0.07100	-0.28643	-0.34052	-0.11530	-0.24158	-0.06111	-0.01196	0.36580	0.18466	0.20800	0.04139
	0.7346	0.7201	0.8455	0.4224	0.3356	0.7511	0.5013	0.8668	0.9738	0.2986	0.6095	0.5699	0.9096
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_D	0.64493	0.75821	0.77537	-0.43968	0.33091	0.60852	0.64374	0.65019	0.35020	0.50559	0.49561	0.55538	0.72112
	0.0441	0.0110	0.0084	0.2036	0.3503	0.0614	0.0446	0.0418	0.3212	0.1360	0.1452	0.0956	0.0186
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_E	0.39898	0.52803	0.52908	-0.29298	0.16286	0.37064	0.42490	0.41941	0.12792	0.26358	0.23391	0.29030	0.47714
	0.2534	0.1167	0.1158	0.4113	0.6535	0.2917	0.2209	0.2276	0.7247	0.4618	0.5154	0.4158	0.1632
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_L	0.64852	0.52956	0.48329	-0.11816	0.42693	0.64683	0.61554	0.59813	0.67086	0.56245	0.62466	0.62513	0.52789
	0.0425	0.1154	0.1570	0.7451	0.2185	0.0432	0.0582	0.0678	0.0337	0.0906	0.0535	0.0533	0.1168
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_VG	0.66810	0.76534	0.84829	-0.38374	0.18283	0.61474	0.67299	0.62795	0.30901	0.44707	0.46349	0.53733	0.70986
	0.0347	0.0099	0.0019	0.2736	0.6132	0.0586	0.0329	0.0519	0.3650	0.1952	0.1773	0.1092	0.0215
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_N	0.75561	0.62882	0.85906	-0.43259	0.31959	0.71443	0.75030	0.73313	0.46169	0.58249	0.59299	0.65517	0.78592
	0.0115	0.0030	0.0015	0.2118	0.3680	0.0203	0.0124	0.0158	0.1792	0.0772	0.0708	0.0398	0.0070
	10	10	10	10	10	10	10	10	10	10	10	10	10
A_NE	0.84753	0.90347	0.93239	-0.41316	0.45333	0.80878	0.83629	0.81800	0.57551	0.66858	0.69387	0.75059	0.86417
	0.0020	0.0003	0.0001	0.2353	0.1882	0.0046	0.0026	0.0038	0.0817	0.0346	0.0260	0.0124	0.0013
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_N1	0.69845	0.80426	0.86737	-0.42360	0.25232	0.64838	0.70069	0.67213	0.35024	0.50328	0.50527	0.57896	0.75271
	0.0247	0.0050	0.0011	0.2225	0.4819	0.0422	0.0240	0.0328	0.3211	0.1361	0.1363	0.0795	0.0120
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_N2	0.71600	0.78780	0.84892	-0.38874	0.22506	0.66764	0.71850	0.67617	0.39548	0.50486	0.52773	0.59395	0.73399
	0.0199	0.0068	0.0020	0.2669	0.5319	0.0349	0.0192	0.0318	0.2580	0.1367	0.1169	0.0702	0.0157
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_BN1	0.59764	0.69937	0.75644	-0.40449	0.11698	0.54939	0.60456	0.57422	0.25594	0.40932	0.40466	0.47787	0.64642
	0.0681	0.0244	0.0113	0.2463	0.7476	0.1000	0.0641	0.0826	0.4764	0.2401	0.2461	0.1624	0.0434
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_BN2	0.64046	0.73227	0.78505	-0.36906	0.15654	0.58282	0.65067	0.61192	0.30339	0.43600	0.43913	0.51125	0.67560
	0.0461	0.0160	0.0071	0.2939	0.6658	0.0709	0.0416	0.0601	0.3941	0.2078	0.2042	0.1310	0.0320
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_V	0.04525	0.07183	0.14939	-0.37994	-0.31856	0.03723	-0.06331	0.09658	0.04136	0.45218	0.27478	0.31658	0.21820
	0.9012	0.8437	0.6804	0.2788	0.3697	0.9187	0.8621	0.7907	0.9097	0.1895	0.4423	0.3696	0.5448
	10	10	10	10	10	10	10	10	10	10	10	10	10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	C_V	C_D	C_E	C_L	C_VG	C_N	C_NE	C1_N1	C1_N2	C1_BN1	C1_BN2	C1_V	C1_D
A1_D	0.89073	0.70952	0.75864	-0.39636	0.17492	0.54559	0.89924	0.57921	0.24586	0.41278	0.39571	0.47068	0.85806
	0.0721	0.0216	0.0110	0.2568	0.6289	0.1028	0.0671	0.0793	0.4936	0.2358	0.2577	0.1698	0.0368
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_E	0.44574	0.56186	0.89080	-0.28441	0.10173	0.41116	0.46830	0.45051	0.14331	0.29439	0.28683	0.32686	0.51083
	0.1967	0.0911	0.0783	0.4258	0.7798	0.2378	0.1743	0.1913	0.6929	0.4090	0.4742	0.3569	0.1312
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_L	0.84810	0.82051	0.84131	-0.26220	0.22765	0.81726	0.83836	0.59079	0.48596	0.47153	0.83614	0.56584	0.58492
	0.0427	0.0656	0.0457	0.4643	0.5267	0.0573	0.0470	0.0721	0.1544	0.1669	0.1101	0.0882	0.0757
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_VG	0.84033	0.73362	0.80139	-0.36508	0.15177	0.58993	0.84869	0.60605	0.29142	0.42322	0.43614	0.50825	0.87784
	0.0461	0.0157	0.0063	0.2996	0.6755	0.0726	0.0425	0.0638	0.4139	0.2230	0.2077	0.1336	0.0312
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_N	0.82843	0.72373	0.77803	-0.38547	0.14518	0.58052	0.83665	0.60168	0.28924	0.42973	0.43059	0.50303	0.86691
	0.0517	0.0180	0.0080	0.2713	0.6880	0.0785	0.0477	0.0657	0.4178	0.2152	0.2142	0.1383	0.0345
	10	10	10	10	10	10	10	10	10	10	10	10	10
A1_NE	0.71345	0.80825	0.88513	-0.40529	0.24404	0.66452	0.71698	0.68187	0.37445	0.50789	0.51976	0.58995	0.75142
	0.0205	0.0048	0.0012	0.2453	0.4968	0.0361	0.0196	0.0299	0.2884	0.1339	0.1244	0.0726	0.0122
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_N1	0.98418	0.87245	0.85786	-0.18847	0.62567	0.87825	0.89651	0.97286	0.94582	0.74874	0.83632	0.84887	0.92448
	0.0001	0.0001	0.0015	0.6001	0.0033	0.0001	0.0001	0.0001	0.0020	0.0127	0.0026	0.0019	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_N2	0.88441	0.79088	0.89055	0.15194	0.86766	0.90686	0.92965	0.87724	0.92017	0.65662	0.75827	0.73884	0.73068
	0.0007	0.0064	0.0723	0.6752	0.0011	0.0003	0.0001	0.0009	0.0002	0.0392	0.0109	0.0146	0.0164
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_BN1	0.99072	0.94802	0.81821	-0.27682	0.82072	0.98463	0.95866	0.98536	0.91427	0.88520	0.93686	0.94663	0.85529
	0.0001	0.0001	0.0038	0.4388	0.0036	0.0001	0.0001	0.0001	0.0002	0.0007	0.0001	0.0001	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_BN2	0.99436	0.95124	0.81462	-0.24278	0.86522	0.98790	0.97807	0.99107	0.91570	0.82719	0.91351	0.91407	0.83778
	0.0001	0.0001	0.0041	0.4991	0.0012	0.0001	0.0001	0.0001	0.0002	0.0031	0.0002	0.0002	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_V	1.00000	0.97289	0.86310	-0.29163	0.82277	0.98631	0.97817	0.99273	0.98367	0.83509	0.90848	0.92104	0.95956
	0.0000	0.0001	0.0013	0.4136	0.0035	0.0001	0.0001	0.0001	0.0007	0.0026	0.0003	0.0002	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_D	0.97289	1.00000	0.84438	-0.38209	0.75158	0.95321	0.95704	0.95837	0.76077	0.76880	0.84163	0.86947	0.97191
	0.0001	0.0000	0.0001	0.2759	0.0122	0.0001	0.0001	0.0001	0.0106	0.0094	0.0023	0.0011	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_E	0.86310	0.84438	1.00000	-0.49176	0.54927	0.81675	0.84088	0.82752	0.64649	0.64933	0.70616	0.75629	0.80817
	0.0013	0.0001	0.0000	0.1488	0.1001	0.0038	0.0023	0.0031	0.1022	0.0422	0.0227	0.0110	0.0003
	10	10	10	10	10	10	10	10	10	10	10	10	10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	C_V	C_D	C_E	C_L	C_VG	C_N	C_NE	C1_N1	C1_N2	C1_BN1	C1_BN2	C1_V	C1_D
C_L	-0.29163	-0.38209	-0.49176	1.00000	-0.05811	-0.25674	-0.13439	-0.30223	-0.05460	-0.42696	-0.41086	-0.44765	-0.49755
	0.4136	0.2759	0.1488	0.0000	0.8733	0.4740	0.7113	0.3960	0.8809	0.2185	0.2382	0.1945	0.1434
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_VG	0.82277	0.75158	0.54927	-0.05811	1.00000	0.85095	0.82318	0.83595	0.86883	0.85357	0.77320	0.73560	0.73414
	0.0035	0.0122	0.1001	0.8733	0.0000	0.0018	0.0034	0.0026	0.0011	0.0404	0.0087	0.0153	0.0156
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_N	0.99631	0.95321	0.81675	-0.25674	0.85095	1.00000	0.97387	0.99608	0.81824	0.85238	0.92561	0.92972	0.94771
	0.0001	0.0001	0.0038	0.4740	0.0018	0.0000	0.0001	0.0001	0.0002	0.0017	0.0001	0.0001	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C_NE	0.97917	0.95704	0.84098	-0.13439	0.82318	0.97387	1.00000	0.96175	0.85388	0.72830	0.82075	0.83163	0.89988
	0.0001	0.0001	0.0023	0.7113	0.0034	0.0001	0.0000	0.0001	0.0017	0.0169	0.0036	0.0029	0.0004
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_N1	0.99273	0.95937	0.82752	-0.30223	0.83595	0.99608	0.96175	1.00000	0.90648	0.87571	0.93799	0.94459	0.96544
	0.0001	0.0001	0.0031	0.3960	0.0026	0.0001	0.0001	0.0000	0.0003	0.0009	0.0001	0.0001	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_N2	0.88367	0.76077	0.54649	-0.05460	0.86883	0.91824	0.85388	0.90648	1.00000	0.86325	0.91885	0.89077	0.79252
	0.0007	0.0106	0.1022	0.8809	0.0011	0.0002	0.0017	0.0003	0.0000	0.0013	0.0002	0.0005	0.0063
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_BN1	0.83509	0.76880	0.64933	-0.42696	0.85357	0.85238	0.72830	0.87571	0.86325	1.00000	0.97408	0.97849	0.88041
	0.0026	0.0094	0.0422	0.2185	0.0404	0.0017	0.0169	0.0009	0.0013	0.0000	0.0001	0.0001	0.0008
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_BN2	0.90948	0.84163	0.70516	-0.41086	0.77320	0.92561	0.82075	0.93799	0.91885	0.97408	1.00000	0.99438	0.92406
	0.0003	0.0023	0.0227	0.2382	0.0087	0.0001	0.0036	0.0001	0.0002	0.0001	0.0000	0.0001	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_V	0.92104	0.86947	0.75829	-0.44765	0.73560	0.92972	0.83163	0.94459	0.89077	0.97849	0.99438	1.00000	0.94763
	0.0002	0.0011	0.0110	0.1945	0.0153	0.0001	0.0029	0.0001	0.0005	0.0001	0.0001	0.0000	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_D	0.95956	0.97191	0.90617	-0.49755	0.73414	0.94771	0.89988	0.96544	0.79252	0.88041	0.92406	0.94763	1.00000
	0.0001	0.0001	0.0003	0.1434	0.0156	0.0001	0.0004	0.0001	0.0063	0.0008	0.0001	0.0001	0.0000
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_E	0.84067	0.91747	0.97897	-0.60782	0.51190	0.79979	0.78042	0.81881	0.54509	0.72586	0.75870	0.51073	0.92803
	0.0023	0.0002	0.0001	0.0623	0.1304	0.0055	0.0077	0.0038	0.1032	0.0175	0.0110	0.0044	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_L	-0.26976	-0.40508	-0.54998	0.94197	-0.01533	-0.21764	-0.15925	-0.25475	0.07767	-0.23660	-0.25925	-0.30831	-0.45374
	0.4510	0.2455	0.0995	0.0001	0.9665	0.5458	0.6603	0.4775	0.8311	0.5105	0.4695	0.3861	0.1878
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_VG	0.77861	0.69180	0.47827	-0.07685	0.99071	0.81319	0.76232	0.79818	0.86111	0.65381	0.76976	0.72471	0.69620
	0.0060	0.0267	0.1620	0.8329	0.0001	0.0042	0.0104	0.0056	0.0014	0.0403	0.0092	0.0177	0.0253
	10	10	10	10	10	10	10	10	10	10	10	10	10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	C_V	C_D	C_E	C_L	C_VG	C_N	C_NE	C1_N1	C1_N2	C1_BN1	C1_BN2	C1_V	C1_D
C1_N	0.88506	0.61686	0.68581	-0.41898	0.73049	0.90192	0.78850	0.91864	0.90289	0.98080	0.98582	0.98409	0.91162
	0.0007	0.0039	0.0266	0.2281	0.0164	0.0004	0.0067	0.0002	0.0003	0.0001	0.0001	0.0001	0.0002
	10	10	10	10	10	10	10	10	10	10	10	10	10
C1_NE	0.99421	0.94932	0.81581	-0.26959	0.84092	0.99881	0.96557	0.98732	0.92411	0.87501	0.93997	0.94444	0.95287
	0.0001	0.0001	0.0040	0.4513	0.0023	0.0001	0.0001	0.0001	0.0001	0.0008	0.0001	0.0001	0.0001
	10	10	10	10	10	10	10	10	10	10	10	10	10

	C1_E	C1_L	C1_VG	C1_N	C1_NE
A_N1	0.89759	-0.47003	0.38006	0.67804	0.76930
	0.0004	0.1704	0.2787	0.0312	0.0066
	10	10	10	10	10
A_N2	0.86531	-0.40193	0.35975	0.69341	0.81266
	0.0012	0.2496	0.3072	0.0262	0.0043
	10	10	10	10	10
A_BN1	0.77626	-0.50983	0.19688	0.56504	0.66429
	0.0083	0.1322	0.5856	0.0888	0.0362
	10	10	10	10	10
A_BN2	0.83863	-0.47388	0.24270	0.59696	0.73199
	0.0024	0.1665	0.4993	0.0684	0.0161
	10	10	10	10	10
A_V	0.06426	-0.03038	-0.32936	0.25778	-0.08730
	0.8600	0.9336	0.3527	0.4721	0.8534
	10	10	10	10	10
A_D	0.74231	-0.49721	0.23922	0.50104	0.61162
	0.0139	0.1437	0.5056	0.1402	0.0602
	10	10	10	10	10
A_E	0.48524	-0.39169	0.07757	0.24567	0.36788
	0.1551	0.2630	0.8313	0.4939	0.2955
	10	10	10	10	10
A_L	0.47455	-0.05044	0.42272	0.60460	0.64685
	0.1658	0.8899	0.2236	0.0641	0.0432
	10	10	10	10	10
A_VG	0.82096	-0.50911	0.10345	0.45845	0.60976
	0.0036	0.1329	0.7761	0.1827	0.0612
	10	10	10	10	10
A_N	0.82499	-0.48680	0.22992	0.59121	0.71587
	0.0033	0.1517	0.5228	0.0719	0.0189
	10	10	10	10	10
A_NE	0.90044	-0.44704	0.36897	0.68690	0.81037

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	C1_E	C1_L	C1_VG	C1_N	C1_NE
A1_N1	0.83442	-0.82219	0.16227	0.50608	0.64759
	0.0027	0.1215	0.6542	0.1356	0.0429
	10	10	10	10	10
A1_N2	0.81222	-0.48329	0.14151	0.52069	0.66523
	0.0043	0.1570	0.6966	0.1228	0.0358
	10	10	10	10	10
A1_BN1	0.72359	-0.51843	0.03066	0.40767	0.54663
	0.0180	0.1239	0.9326	0.2422	0.1020
	10	10	10	10	10
A1_BN2	0.74898	-0.48710	0.07219	0.43932	0.58887
	0.0125	0.1533	0.8429	0.2040	0.0733
	10	10	10	10	10
A1_V	0.26951	-0.18050	-0.33239	0.34657	0.08143
	0.4514	0.6178	0.3480	0.3266	0.8231
	10	10	10	10	10
A1_D	0.72501	-0.51123	0.08834	0.40346	0.54233
	0.0177	0.1310	0.8082	0.2476	0.1053
	10	10	10	10	10
A1_E	0.54585	-0.40982	0.02655	0.27199	0.40558
	0.1026	0.2395	0.9420	0.4471	0.2449
	10	10	10	10	10
A1_L	0.61781	-0.29361	0.18106	0.51355	0.61567
	0.0570	0.4103	0.6167	0.1289	0.0581
	10	10	10	10	10
A1_VG	0.77298	-0.49502	0.07387	0.43241	0.58442
	0.0087	0.1458	0.8393	0.2120	0.0760
	10	10	10	10	10
A1_N	0.74415	-0.50190	0.06028	0.43161	0.57702
	0.0136	0.1394	0.8686	0.2130	0.0807
	10	10	10	10	10
A1_NE	0.83082	-0.50458	0.15590	0.51626	0.66222
	0.0029	0.1368	0.6671	0.1266	0.0370
	10	10	10	10	10
C_N1	0.90580	-0.21138	0.76596	0.80600	0.97104
	0.0049	0.5577	0.0098	0.0049	0.0001
	10	10	10	10	10
C_N2	0.51578	0.16256	0.82585	0.72335	0.89617
	0.1270	0.6536	0.0032	0.0181	0.0004
	10	10	10	10	10

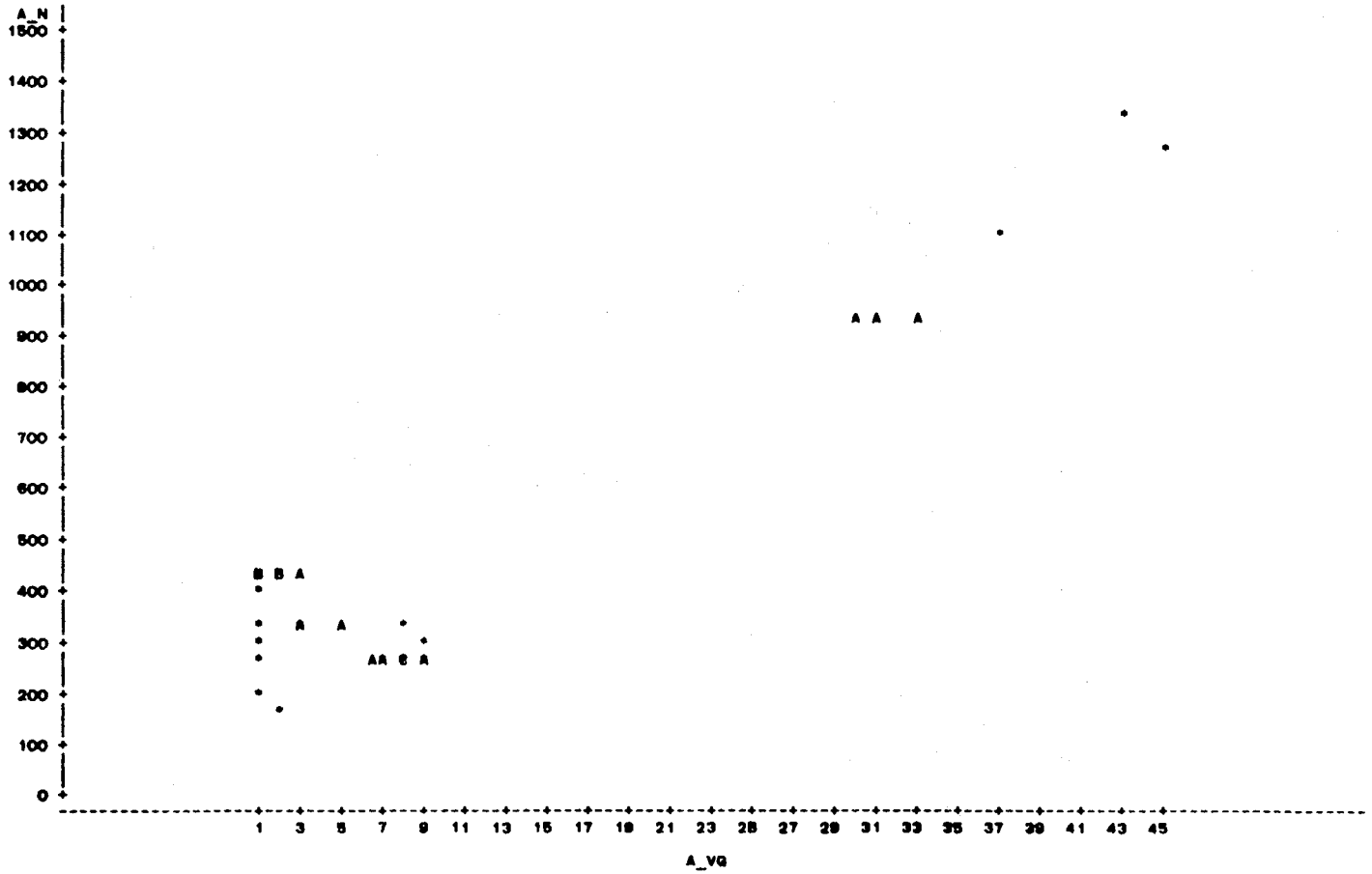
PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	C1_E	C1_L	C1_VG	C1_N	C1_NE
C_BN1	0.80847	-0.22323	0.78510	0.92184	0.99643
	0.0045	0.5353	0.0071	0.0001	0.0001
	10	10	10	10	10
C_BN2	0.78937	-0.21296	0.82832	0.88453	0.99484
	0.0066	0.5547	0.0032	0.0007	0.0001
	10	10	10	10	10
C_V	0.84067	-0.26976	0.77861	0.88506	0.99421
	0.0023	0.4510	0.0080	0.0007	0.0001
	10	10	10	10	10
C_D	0.91747	-0.40508	0.69180	0.81686	0.94932
	0.0002	0.2455	0.0267	0.0039	0.0001
	10	10	10	10	10
C_E	0.97897	-0.54998	0.47827	0.68581	0.81581
	0.0001	0.0995	0.1620	0.0286	0.0040
	10	10	10	10	10
C_L	-0.60782	0.94197	-0.07885	-0.41898	-0.26959
	0.0623	0.0001	0.8329	0.2281	0.4513
	10	10	10	10	10
C_VG	0.51180	-0.01533	0.99071	0.73049	0.84092
	0.1304	0.9665	0.0001	0.0164	0.0023
	10	10	10	10	10
C_N	0.79979	-0.21764	0.81319	0.90192	0.99881
	0.0055	0.5458	0.0042	0.0004	0.0001
	10	10	10	10	10
C_NE	0.78042	-0.15925	0.76232	0.78850	0.96557
	0.0077	0.6603	0.0104	0.0067	0.0001
	10	10	10	10	10
C1_N1	0.81881	-0.25475	0.79818	0.91864	0.99732
	0.0038	0.4775	0.0056	0.0002	0.0001
	10	10	10	10	10
C1_N2	0.54509	0.07767	0.86111	0.90298	0.92411
	0.1032	0.8311	0.0014	0.0003	0.0001
	10	10	10	10	10
C1_BN1	0.72586	-0.23660	0.65381	0.99050	0.67501
	0.0175	0.5105	0.0403	0.0001	0.0009
	10	10	10	10	10
C1_BN2	0.75870	-0.25925	0.76976	0.99592	0.93997
	0.0110	0.4695	0.0092	0.0001	0.0001
	10	10	10	10	10

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

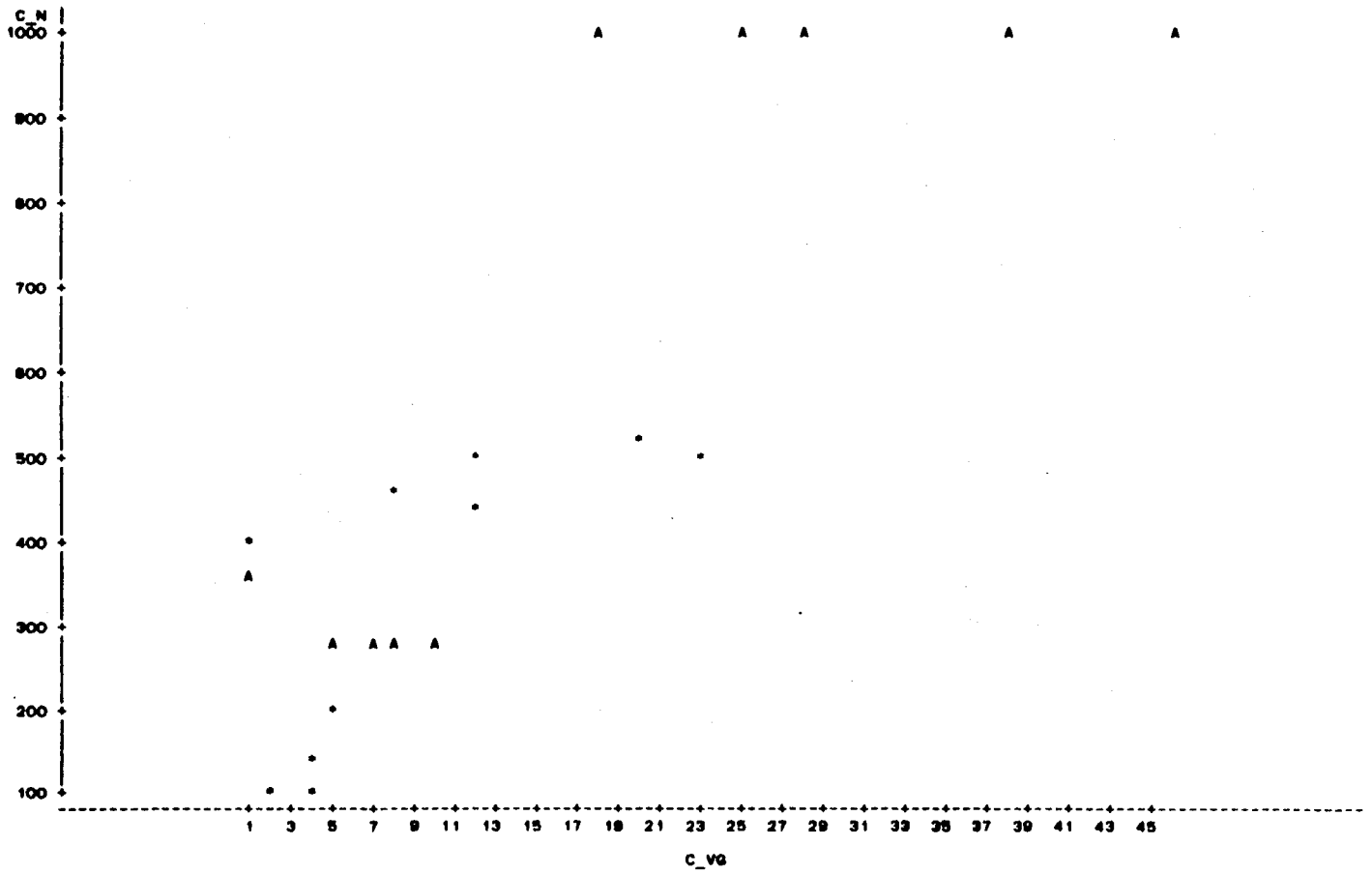
	C1_E	C1_L	C1_VG	C1_N	C1_NE
C1_V	0.81073	-0.30831	0.72471	0.89409	0.94444
	0.0044	0.3861	0.0177	0.0001	0.0001
	10	10	10	10	10
C1_D	0.92803	-0.45374	0.69620	0.91162	0.95287
	0.0001	0.1878	0.0253	0.0002	0.0001
	10	10	10	10	10
C1_E	1.00000	-0.61934	0.45976	0.74894	0.80380
	0.0000	0.0562	0.1813	0.0127	0.0051
	10	10	10	10	10
C1_L	-0.61934	1.00000	-0.01242	-0.25061	-0.21644
	0.0562	0.0000	0.9728	0.4849	0.5481
	10	10	10	10	10
C1_VG	0.45976	-0.01242	1.00000	0.72875	0.80427
	0.1813	0.9728	0.0000	0.0168	0.0050
	10	10	10	10	10
C1_N	0.74894	-0.25061	0.72875	1.00000	0.91966
	0.0127	0.4849	0.0168	0.0000	0.0002
	10	10	10	10	10
C1_NE	0.80380	-0.21644	0.80427	0.91966	1.00000
	0.0051	0.5481	0.0050	0.0002	0.0000
	10	10	10	10	10

PLOT OF A_N*A_VQ LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF A1_N*A1_VQ SYMBOL USED IS *



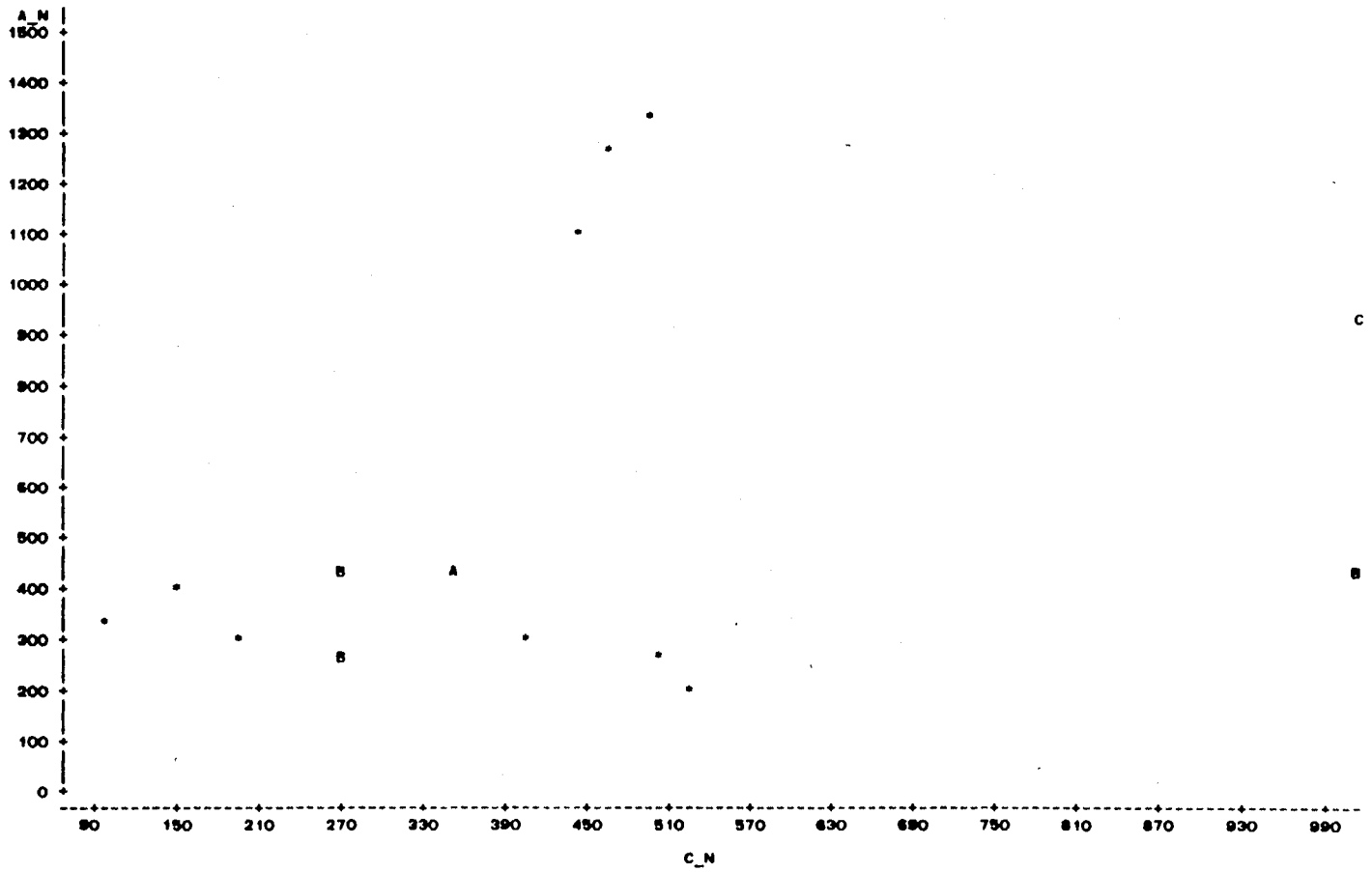
NOTE: 1 OBS HIDDEN

PLOT OF C_N*C_VG LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF C1_N*C1_VG SYMBOL USED IS *



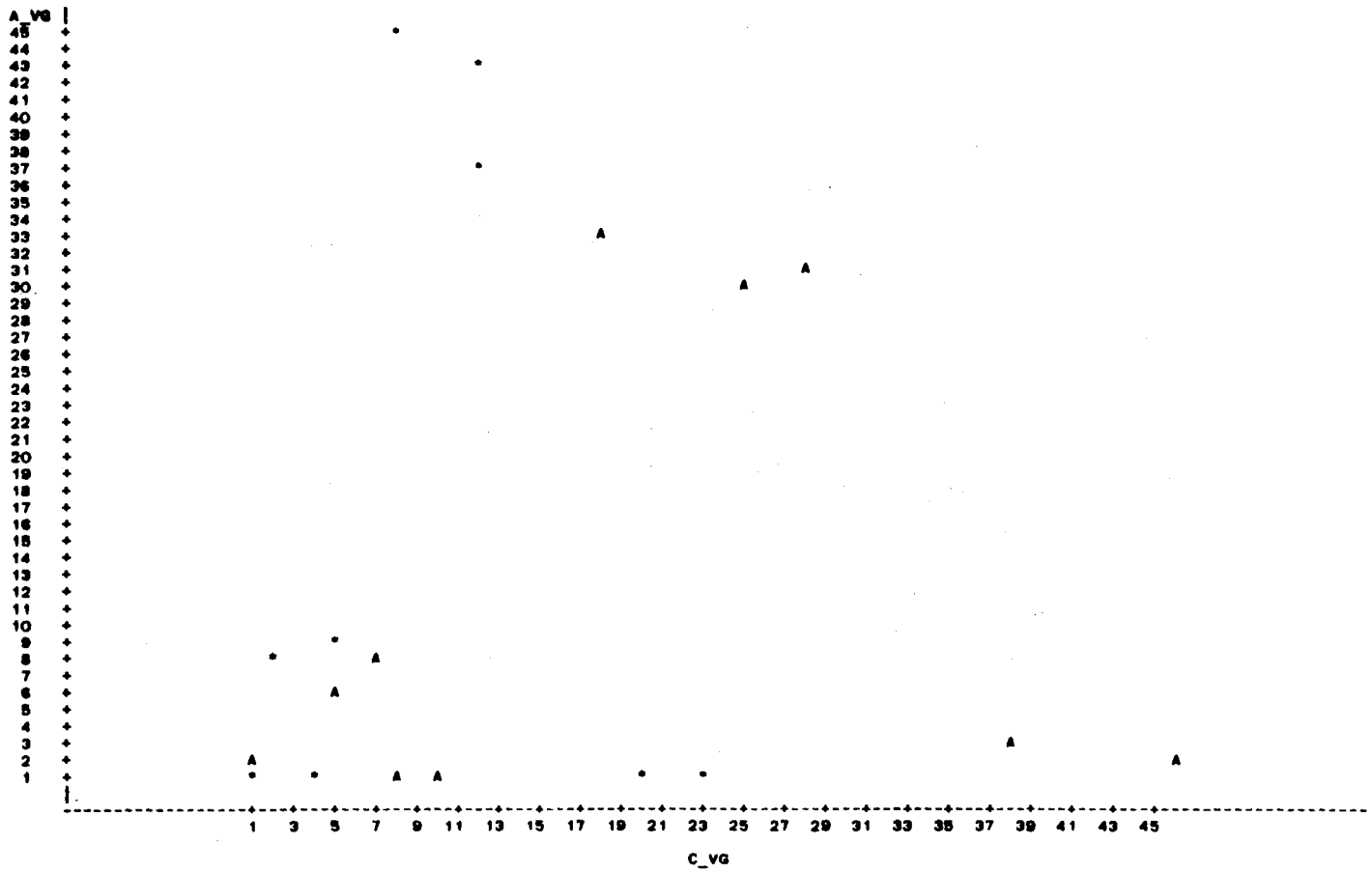
NOTE: 12 OBS HAD MISSING VALUES

PLOT OF A_N*C_N LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF A1_N*C1_N SYMBOL USED IS *



NOTE: 12 OBS HAD MISSING VALUES 1 OBS HIDDEN

PLOT OF A_VG*C_VG LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF A1_VG*C1_VG SYMBOL USED IS *



NOTE: 12 OBS HAD MISSING VALUES 1 OBS HIDDEN

The following tables and plots are for the expert Ada programs and novice Ada programs analysis and comparison. The novice metrics are extrapolated with N. The variables are represented as

g_m.

Where g could be E for expert; or

N for novice;

and

m is the selected metric. (see nomenclature)

OBS	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE
1	101	76	330	603	4829	401	1936426	0.015	35	833	795
2	101	76	330	603	4829	401	1936426	0.015	35	833	795
3	101	76	330	603	4828	401	1936426	0.015	35	833	795
4	26	24	182	257	1756	139	244519	0.081	1	449	161
5	26	24	182	257	1756	139	244518	0.081	1	449	161
6	26	24	182	257	1756	139	244519	0.081	1	449	161
7	26	24	182	257	1756	139	244519	0.081	1	449	161
8	26	24	182	257	1756	139	244519	0.081	1	449	161
9	17	26	109	172	1067	56	59430	0.334	0	281	133
10	17	26	109	172	1067	56	59430	0.334	0	281	133
11	17	26	109	172	1067	56	59430	0.334	0	281	133
12	17	26	109	172	1067	56	59430	0.334	0	281	133
13	17	26	109	172	1067	56	59430	0.334	0	281	133
14	17	26	109	172	1067	56	59430	0.334	0	281	133
15	28	30	112	214	1329	103	137492	0.124	3	326	200
16	28	30	112	214	1329	103	137492	0.124	3	326	200

OBS	N_N1	N_N2	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
1	83	83	325	608	3756	311	386832	0.354	30.00	933	552
2	101	105	321	612	3706	294	281797	0.658	31.00	933	674
3	75	112	315	618	3550	207	176395	1.451	33.00	933	585
4	46	36	181	268	1756	171	181080	0.149	2.00	449	267
5	45	68	158	291	1632	92	52194	1.606	3.00	449	336
6	39	35	191	258	16989	142	142360	0.236	2.00	449	230
7	36	31	190	259	1756	151	198979	0.134	1.00	449	216
8	33	28	192	257	1698	155	189619	0.124	1.00	449	188
9	15	16	98	183	1082	74	112724	0.099	6.37	281	106
10	19	20	108	173	1063	79	94661	0.133	9.00	281	121
11	20	24	97	184	1069	73	80438	0.186	8.00	281	137
12	23	22	111	170	1099	92	112538	0.108	7.00	281	145
13	18	23	112	169	1043	66	67783	0.244	8.00	281	126
14	21	21	1101	170	1075	65	98978	0.120	9.00	281	133
15	70	70	130	196	1108	98	23286	2.515	5.00	326	377
16	28	37	118	208	1202	76	56886	0.535	3.00	326	195

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
E_N1	16	37.0625000	32.0634137	593.000000	17.00000000	101.000000
E_N2	16	35.2500000	20.3027092	564.000000	24.00000000	76.000000
E_BN1	16	176.7500000	84.6738842	2828.000000	109.00000000	330.000000
E_BN2	16	284.6250000	162.0670129	4554.000000	172.00000000	603.000000
E_V	16	2020.4375000	1424.1754559	32327.000000	1067.00000000	4829.000000
E_D	16	152.5000000	128.3074433	2440.000000	56.00000000	401.000000
E_E	16	478964.8125000	727400.2864517	7663437.000000	59430.00000000	1936426.000000
E_L	16	1.7200000000E-01	1.3387158036-01	2.752000	1.5000000000E-02	3.3400000000E-01
E_VG	16	10.2500000	12.6517456	164.000000	1.0000000000E+00	35.000000
E_N	16	461.3750000	244.9380534	7382.000000	281.00000000	933.000000
E_NE	16	274.2500000	259.2817001	4388.000000	133.00000000	795.000000
N_N1	16	42.0000000	26.4600328	672.000000	15.00000000	101.000000
N_N2	16	45.8125000	31.4011014	733.000000	18.00000000	112.000000
N_BN1	16	234.2500000	244.3575250	3748.000000	97.00000000	1101.000000
N_BN2	16	289.0000000	165.5918678	4624.000000	169.00000000	618.000000
N_V	16	2723.9375000	3929.6031262	43583.000000	1043.00000000	16989.000000
N_D	16	135.3750000	77.4380397	2166.000000	66.00000000	311.000000
N_E	16	141034.3750000	93884.0037989	2256550.000000	23286.00000000	386832.000000
N_L	16	5.4075000000E-01	7.0402239556-01	8.652000	9.9000000000E-02	2.515000
N_VG	16	9.8356250	11.0083138	157.370000	1.0000000000E+00	33.000000
N_N	16	461.3750000	244.9380534	7382.000000	281.00000000	933.000000
N_NE	16	274.1875000	181.9539204	4387.000000	105.00000000	674.000000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 16

	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE	N_N1	N_N2
E_N1	1.00000 0.0000	0.98691 0.0001	0.92700 0.0001	0.99161 0.0001	0.99169 0.0001	0.98569 0.0001	0.99626 0.0001	-0.68899 0.0032	0.92800 0.0001	0.97657 0.0001	0.99718 0.0001	0.88331 0.0001	0.99788 0.0001
E_N2	0.98691 0.0001	1.00000 0.0000	0.88545 0.0001	0.96174 0.0001	0.96467 0.0001	0.94726 0.0001	0.98514 0.0001	-0.56719 0.0219	0.97250 0.0001	0.93553 0.0001	0.99606 0.0001	0.83493 0.0001	0.96289 0.0001
E_BN1	0.92700 0.0001	0.88545 0.0001	1.00000 0.0000	0.96770 0.0001	0.96846 0.0001	0.97479 0.0001	0.93839 0.0001	-0.81158 0.0001	0.77957 0.0004	0.98599 0.0001	0.90434 0.0001	0.84969 0.0001	0.93731 0.0001
E_BN2	0.99161 0.0001	0.96174 0.0001	0.96770 0.0001	1.00000 0.0000	0.99966 0.0001	0.99839 0.0001	0.99313 0.0001	-0.74184 0.0010	0.89287 0.0001	0.99620 0.0001	0.98189 0.0001	0.88576 0.0001	0.99171 0.0001
E_V	0.99169 0.0001	0.96467 0.0001	0.96846 0.0001	0.99966 0.0001	1.00000 0.0000	0.99677 0.0001	0.99505 0.0001	-0.72595 0.0015	0.90154 0.0001	0.99554 0.0001	0.98361 0.0001	0.87925 0.0001	0.99577 0.0001
E_D	0.98569 0.0001	0.94726 0.0001	0.97479 0.0001	0.99839 0.0001	0.99677 0.0001	1.00000 0.0000	0.98508 0.0001	-0.77866 0.0004	0.86628 0.0001	0.99758 0.0001	0.97191 0.0001	0.89602 0.0001	0.99730 0.0001
E_E	0.99626 0.0001	0.98514 0.0001	0.93839 0.0001	0.99313 0.0001	0.99505 0.0001	0.98508 0.0001	1.00000 0.0000	-0.66169 0.0052	0.93838 0.0001	0.98152 0.0001	0.99537 0.0001	0.86142 0.0001	0.97730 0.0001
E_L	-0.68899 0.0032	-0.56719 0.0219	-0.81158 0.0001	-0.74184 0.0010	-0.72595 0.0015	-0.77866 0.0004	-0.66169 0.0052	1.00000 0.0000	-0.37008 0.1583	-0.77147 0.0005	-0.63300 0.0085	-0.80352 0.0002	-0.75390 0.0007
E_VG	0.92800 0.0001	0.97250 0.0001	0.77957 0.0004	0.89287 0.0001	0.90154 0.0001	0.86838 0.0001	0.93838 0.0001	-0.37008 0.1583	1.00000 0.0000	0.86027 0.0001	0.96288 0.0001	0.71852 0.0018	0.76013 0.0006
E_N	0.97657 0.0001	0.93553 0.0001	0.98599 0.0001	0.99620 0.0001	0.99554 0.0001	0.99758 0.0001	0.98152 0.0001	-0.77147 0.0005	0.86027 0.0001	1.00000 0.0000	0.98237 0.0001	0.87946 0.0001	0.97947 0.0001
E_NE	0.99718 0.0001	0.99606 0.0001	0.90434 0.0001	0.98189 0.0001	0.98361 0.0001	0.97191 0.0001	0.99537 0.0001	-0.63300 0.0085	0.95288 0.0001	0.98237 0.0001	1.00000 0.0000	0.86169 0.0001	0.98250 0.0001
N_N1	0.88331 0.0001	0.83493 0.0001	0.84969 0.0001	0.88576 0.0001	0.87925 0.0001	0.89602 0.0001	0.86142 0.0001	-0.80352 0.0002	0.71852 0.0018	0.87946 0.0001	0.86169 0.0001	1.00000 0.0000	0.93524 0.0001
N_N2	0.99788 0.0001	0.96289 0.0001	0.93731 0.0001	0.99171 0.0001	0.99577 0.0001	0.99730 0.0001	0.97730 0.0001	-0.75390 0.0007	0.76013 0.0006	0.97947 0.0001	0.98250 0.0001	0.93524 0.0001	1.00000 0.0000
N_BN1	0.14222 0.5993	0.16883 0.5324	0.11243 0.6785	0.13469 0.6190	0.14083 0.6029	0.12166 0.6536	0.15735 0.5606	0.05486 0.8401	0.21327 0.4277	0.12788 0.6367	0.15671 0.5622	0.07446 0.7840	0.08362 0.8149
N_BN2	0.98526 0.0001	0.95337 0.0001	0.97363 0.0001	0.99781 0.0001	0.99793 0.0001	0.99630 0.0001	0.98975 0.0001	-0.72978 0.0011	0.88689 0.0001	0.99679 0.0001	0.97513 0.0001	0.87466 0.0001	0.99205 0.0001
N_V	0.15469 0.5668	0.09107 0.7373	0.29364 0.2697	0.20516 0.4459	0.20125 0.4548	0.22206 0.4064	0.16266 0.5472	-0.34573 0.1897	0.02798 0.9161	0.23726 0.3763	0.12783 0.6368	0.16812 0.4854	0.12517 0.6442
N_D	0.89249 0.0001	0.84122 0.0001	0.83784 0.0001	0.92307 0.0001	0.92189 0.0001	0.92603 0.0001	0.90016 0.0001	-0.75528 0.0007	0.76236 0.0006	0.93497 0.0001	0.87379 0.0001	0.85879 0.0001	0.73210 0.0013

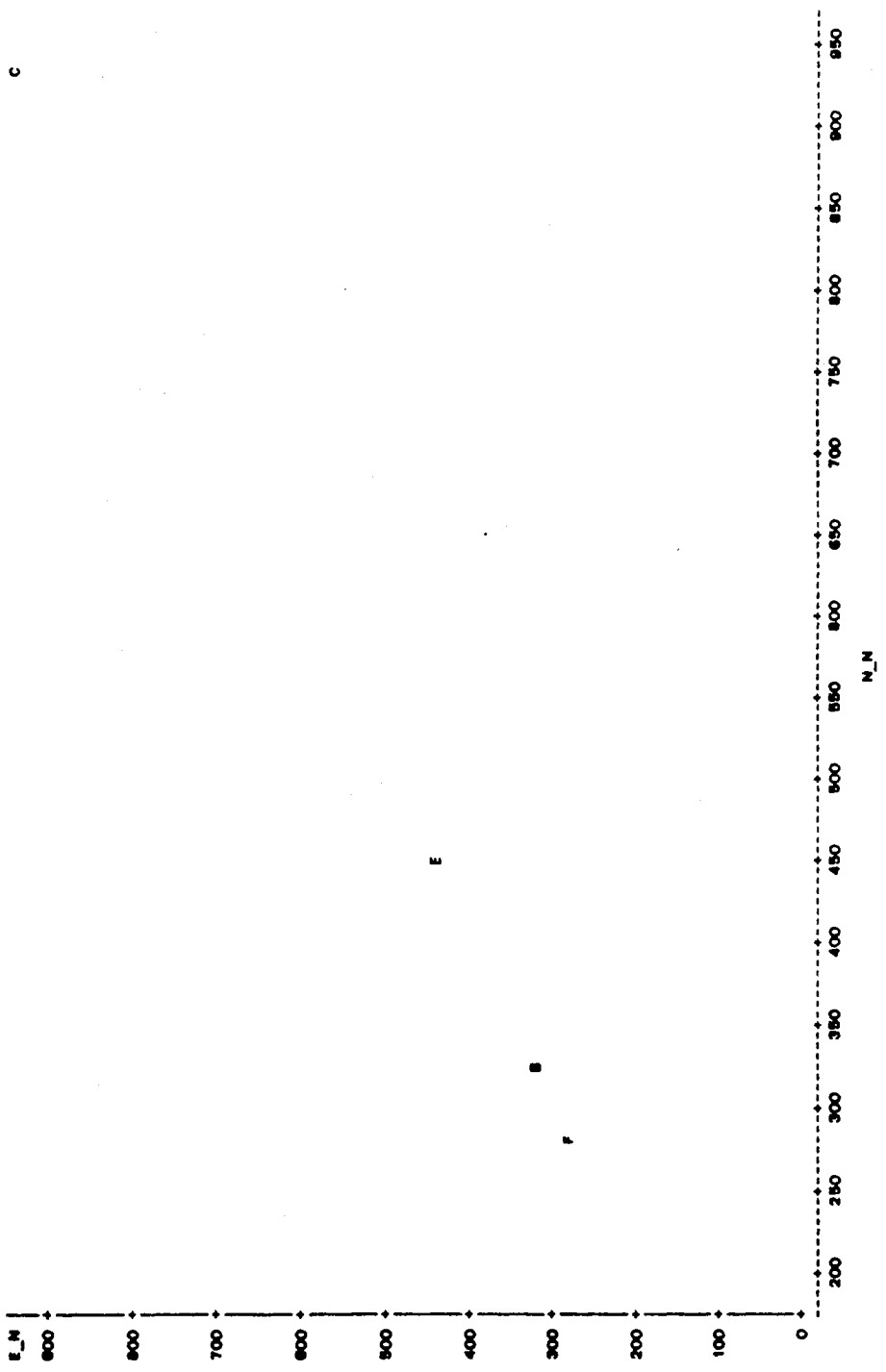
PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 18

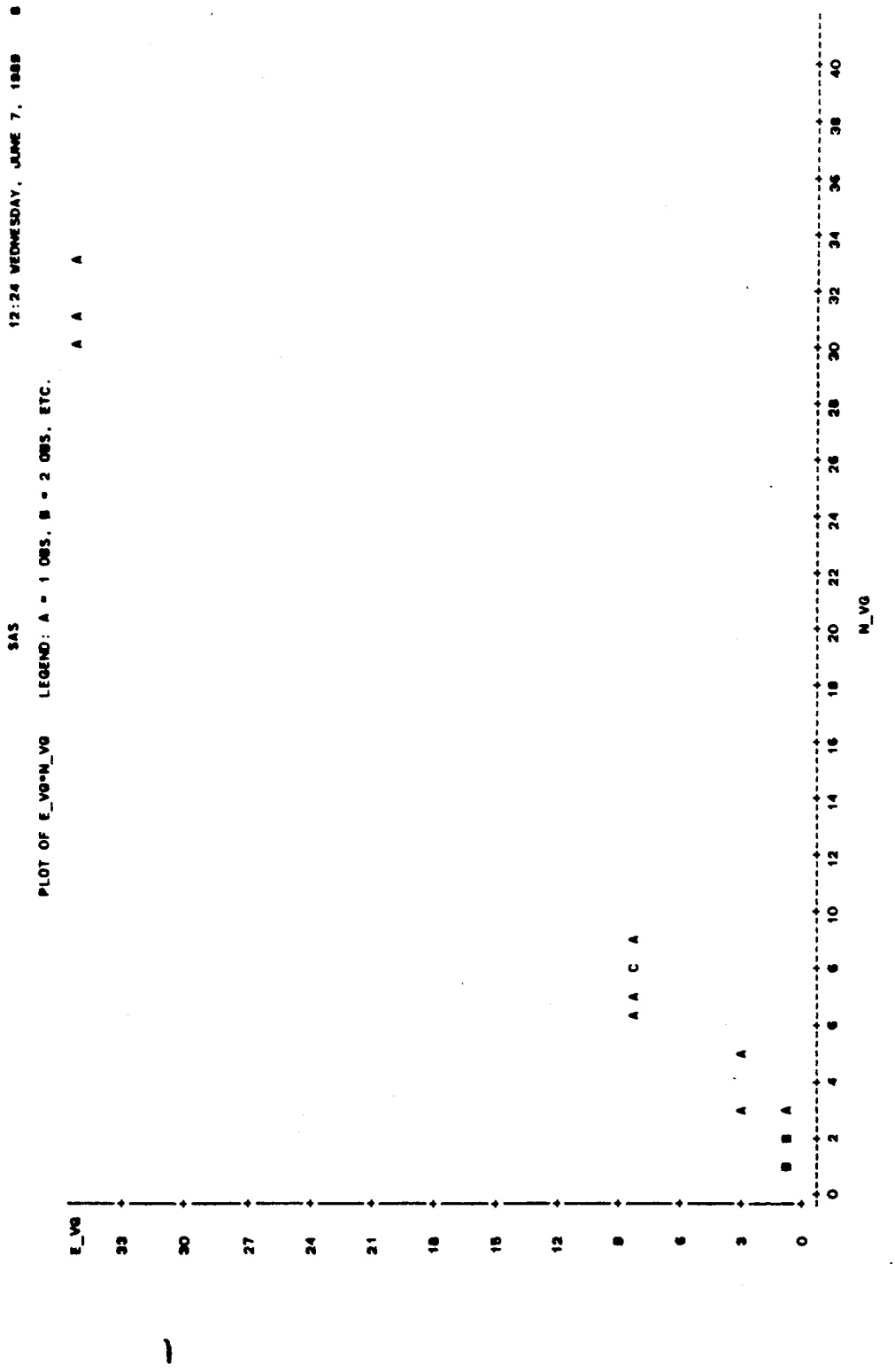
	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE	N_N1	N_N2
N_E	0.74302 0.0010	0.70547 0.0023	0.81800 0.0001	0.78161 0.0003	0.78533 0.0003	0.78021 0.0004	0.76839 0.0005	-0.56515 0.0225	0.87284 0.0043	0.79994 0.0002	0.73243 0.0013	0.81408 0.0114	0.45913 0.0736
N_L	0.26625 0.3189	0.23703 0.3767	0.17219 0.5237	0.23745 0.3759	0.22376 0.4048	0.25450 0.3415	0.21448 0.4250	-0.40395 0.1207	0.12006 0.6578	0.21683 0.4203	0.24679 0.3568	0.52494 0.0368	0.63428 0.0083
N_VG	0.92847 0.0001	0.97192 0.0001	0.77838 0.0004	0.89275 0.0001	0.90099 0.0001	0.86690 0.0001	0.93834 0.0001	-0.37814 0.1487	0.99608 0.0001	0.85979 0.0001	0.95267 0.0001	0.73149 0.0013	0.78802 0.0003
N_N	0.97657 0.0001	0.93553 0.0001	0.98599 0.0001	0.99620 0.0001	0.99554 0.0001	0.99758 0.0001	0.98152 0.0001	-0.77147 0.0005	0.96027 0.0001	1.00000 0.0001	0.96237 0.0001	0.87946 0.0001	0.87947 0.0001
N_NE	0.93825 0.0001	0.89843 0.0001	0.88744 0.0001	0.93962 0.0001	0.93450 0.0001	0.94521 0.0001	0.92296 0.0001	-0.78748 0.0003	0.79661 0.0002	0.93196 0.0001	0.92160 0.0001	0.97930 0.0001	0.97640 0.0001

	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
E_N1	0.14222 0.5993	0.86526 0.0001	0.15489 0.5668	0.89248 0.0001	0.74302 0.0010	0.26625 0.3189	0.92847 0.0001	0.97657 0.0001	0.93825 0.0001
E_N2	0.16863 0.5324	0.95337 0.0001	0.09107 0.7373	0.84122 0.0001	0.70547 0.0023	0.23703 0.3767	0.97192 0.0001	0.93553 0.0001	0.89843 0.0001
E_BN1	0.11243 0.6785	0.87363 0.0001	0.29364 0.2697	0.93784 0.0001	0.81800 0.0001	0.17219 0.5237	0.77838 0.0004	0.98599 0.0001	0.88744 0.0001
E_BN2	0.13469 0.6190	0.89781 0.0001	0.20516 0.4458	0.92307 0.0001	0.78161 0.0003	0.23745 0.3759	0.89275 0.0001	0.99620 0.0001	0.93962 0.0001
E_V	0.14083 0.6028	0.89793 0.0001	0.20125 0.4548	0.92189 0.0001	0.78533 0.0003	0.22376 0.4048	0.90099 0.0001	0.99554 0.0001	0.93450 0.0001
E_D	0.12166 0.6536	0.89630 0.0001	0.22208 0.4084	0.92803 0.0001	0.78021 0.0004	0.25450 0.3415	0.86690 0.0001	0.99758 0.0001	0.94521 0.0001
E_E	0.15735 0.5606	0.88875 0.0001	0.16266 0.5472	0.90018 0.0001	0.76839 0.0005	0.21448 0.4250	0.93834 0.0001	0.98152 0.0001	0.92296 0.0001
E_L	0.05488 0.8401	-0.73978 0.0011	-0.34573 0.1887	-0.75528 0.0007	-0.56515 0.0225	-0.40395 0.1207	-0.37814 0.1487	-0.77147 0.0005	-0.78749 0.0003
E_VG	0.21327 0.4277	0.88689 0.0001	0.02788 0.9181	0.78236 0.0006	0.87284 0.0043	0.12006 0.6578	0.99608 0.0001	0.96027 0.0001	0.79661 0.0002
E_N	0.12798 0.9367	0.89679 0.0001	0.23728 0.3763	0.93497 0.0001	0.79994 0.0002	0.21683 0.4203	0.85979 0.0001	1.00000 0.0001	0.93196 0.0001
E_NE	0.15671 0.5622	0.87513 0.0001	0.12792 0.6369	0.87378 0.0001	0.73243 0.0013	0.24679 0.3568	0.95267 0.0001	0.96237 0.0001	0.92160 0.0001
N_N1	0.07446 0.7840	0.87466 0.0001	0.18812 0.4854	0.85579 0.0001	0.81408 0.0114	0.52494 0.0368	0.73149 0.0013	0.87946 0.0001	0.97930 0.0001

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 16

	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
N_N2	0.06362 0.8149	0.89205 0.0001	0.12517 0.6442	0.73210 0.0013	0.45913 0.0736	0.63428 0.0083	0.78902 0.0003	0.87847 0.0001	0.97640 0.0001
N_BN1	1.00000 0.0000	0.12740 0.6382	-0.01167 0.9658	0.14124 0.6018	0.15530 0.5658	-0.10258 0.7054	0.23896 0.3707	0.12798 0.6367	0.09021 0.7397
N_BN2	0.12740 0.6382	1.00000 0.0000	0.20091 0.4556	0.91527 0.0001	0.77554 0.0004	0.23362 0.3839	0.88741 0.0001	0.99679 0.0001	0.93613 0.0001
N_V	-0.01167 0.9658	0.20091 0.4556	1.00000 0.0000	0.25999 0.3308	0.21199 0.4306	-0.07065 0.7949	0.03185 0.9068	0.23726 0.3763	0.16737 0.5355
N_D	0.14124 0.6018	0.91527 0.0001	0.25999 0.3308	1.00000 0.0000	0.92746 0.0001	0.04111 0.8799	0.74462 0.0009	0.93497 0.0001	0.85201 0.0001
N_E	0.15530 0.5658	0.77554 0.0004	0.21199 0.4306	0.92746 0.0001	1.00000 0.0000	-0.28147 0.2909	0.63125 0.0087	0.79994 0.0002	0.61426 0.0114
N_L	-0.10258 0.7054	0.23362 0.3839	-0.07065 0.7949	0.04111 0.8799	-0.28147 0.2909	1.00000 0.0000	0.17892 0.5073	0.21663 0.4203	0.50389 0.0466
N_VG	0.23896 0.3707	0.88741 0.0001	0.03185 0.9068	0.74462 0.0009	0.63125 0.0087	0.17892 0.5073	1.00000 0.0000	0.85979 0.0001	0.81341 0.0001
N_N	0.12798 0.6367	0.99679 0.0001	0.23726 0.3763	0.93497 0.0001	0.79994 0.0002	0.21663 0.4203	0.85979 0.0001	1.00000 0.0000	0.93196 0.0001
N_NE	0.09021 0.7397	0.93613 0.0001	0.16737 0.5355	0.85201 0.0001	0.61426 0.0114	0.50389 0.0466	0.81341 0.0001	0.93196 0.0001	1.00000 0.0000


SAS 12:24 WEDNESDAY, JUNE 7, 1988 7
PLOT OF E_N*N_N LEGEND: A = 1 OBS, B = 2 OBS, ETC.
E_N
N_N
0
100
200
300
400
500
600
700
800
900
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
C
E
B
F


SAS 12:24 WEDNESDAY, JUNE 7, 1989 8
PLOT OF E_VG*N_VG LEGEND: A = 1 OBS, B = 2 OBS, ETC.
E_VG
N_VG

The following tables and plots are for the expert Ada programs and novice Ada programs analysis and comparison. The novice metrics are extrapolated with Ne. The variables are represented as

g_m.

Where g could be E for expert; or

N for novice;

and

m is the selected metric. (see nomenclature)

OBS	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE
1	101	76	330	603	4829	401	1936426	0.015	35	933	795
2	101	76	330	603	4829	401	1936426	0.015	35	933	795
3	101	76	330	603	4829	401	1936426	0.015	35	933	795
4	26	24	192	257	1756	139	244519	0.091	1	449	161
5	26	24	192	257	1756	139	244519	0.091	1	449	161
6	26	24	192	257	1756	139	244519	0.091	1	449	161
7	26	24	192	257	1756	139	244519	0.091	1	449	161
8	26	24	192	257	1756	139	244519	0.091	1	449	161
9	17	26	109	172	1067	56	59430	0.334	8	281	133
10	17	26	109	172	1067	56	59430	0.334	8	281	133
11	17	26	109	172	1067	56	59430	0.334	8	281	133
12	17	26	109	172	1067	56	59430	0.334	8	281	133
13	17	26	109	172	1067	56	59430	0.334	8	281	133
14	17	26	109	172	1067	56	59430	0.334	8	281	133
15	29	30	112	214	1329	103	137492	0.124	3	326	200
16	29	30	112	214	1328	103	137492	0.124	3	326	200

OBS	N_N1	N_N2	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
1	119	119	468	876	5408	434	556997	0.510	43	1343	795
2	119	123	379	722	4373	347	332575	0.777	37	1101	795
3	101	152	429	840	4826	282	239848	1.970	45	1269	795
4	28	22	109	162	1060	103	109293	0.090	1	271	161
5	21	33	76	140	782	44	25009	0.769	1	215	161
6	27	25	134	181	11915	99	99841	0.166	1	315	161
7	27	23	142	193	1310	113	148459	0.100	1	335	161
8	28	24	165	220	1455	133	162482	0.107	1	385	161
9	19	23	124	230	1365	93	142157	0.125	8	345	133
10	21	22	118	191	1169	87	104104	0.147	9	309	133
11	19	24	94	179	1037	71	78048	0.181	8	273	133
12	21	20	102	156	1011	84	103553	0.099	7	259	133
13	19	25	118	178	1098	70	71409	0.257	9	296	133
14	21	21	111	170	1074	85	98899	0.120	9	281	133
15	37	37	69	104	588	52	12346	1.330	2	173	200
16	28	38	122	213	1235	78	58457	0.550	3	335	200

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
E_N1	16	37.0625000	32.0634137	593.000000	17.00000000	101.000000
E_N2	16	35.2500000	20.3027092	564.000000	24.00000000	76.000000
E_SN1	16	176.7500000	84.6738842	2828.000000	109.00000000	330.000000
E_SN2	16	284.6250000	182.0870128	4554.000000	172.00000000	603.000000
E_V	16	2020.4375000	1424.1784559	32327.000000	1067.00000000	4829.000000
E_D	16	152.5000000	128.3074433	2440.000000	56.00000000	401.000000
E_E	16	478964.8125000	727400.2864517	7663437.000000	59430.00000000	1936426.000000
E_L	16	1.720000000E-01	1.338715803E-01	2.752000	1.500000000E-02	3.340000000E-01
E_VG	16	10.2500000	12.6517456	164.000000	1.000000000E+00	25.000000
E_N	16	461.3750000	244.9380534	7382.000000	281.00000000	933.000000
E_NE	16	274.2500000	259.2817001	4388.000000	133.00000000	795.000000
N_N1	16	40.9375000	36.2720623	655.000000	19.00000000	119.000000
N_N2	16	45.6875000	43.3339262	731.000000	20.00000000	152.000000
N_SN1	16	172.5000000	128.5918608	2760.000000	69.00000000	468.000000
N_SN2	16	297.1875000	259.2188256	4755.000000	104.00000000	876.000000
N_V	16	2481.6250000	2946.3793120	39706.000000	588.00000000	11915.000000
N_D	16	135.9375000	113.8172031	2175.000000	44.00000000	434.000000
N_E	16	148467.3125000	134765.4890253	2343477.000000	12346.00000000	556997.000000
N_L	16	4.561250000E-01	5.351660023E-01	7.298000	9.000000000E-02	1.970000
N_VG	16	11.5625000	15.3664848	185.000000	1.000000000E+00	45.000000
N_N	16	469.0625000	387.3056446	7505.000000	173.00000000	1343.000000
N_NE	16	274.2500000	259.2817001	4388.000000	133.00000000	795.000000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 16

	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE	N_N1	N_N2
E_N1	1.00000 0.0000	0.98681 0.0001	0.92700 0.0001	0.98161 0.0001	0.99169 0.0001	0.98569 0.0001	0.99626 0.0001	-0.68899 0.0032	0.92800 0.0001	0.97887 0.0001	0.98719 0.0001	0.89134 0.0001	0.86233 0.0001
E_N2	0.98681 0.0001	1.00000 0.0000	0.86545 0.0001	0.96174 0.0001	0.96467 0.0001	0.94726 0.0001	0.98514 0.0001	-0.56719 0.0219	0.87250 0.0001	0.93883 0.0001	0.98606 0.0001	0.98578 0.0001	0.98388 0.0001
E_BN1	0.92700 0.0001	0.86545 0.0001	1.00000 0.0000	0.96770 0.0001	0.96646 0.0001	0.97479 0.0001	0.93839 0.0001	-0.81158 0.0001	0.77857 0.0004	0.88588 0.0001	0.90434 0.0001	0.90322 0.0001	0.87783 0.0001
E_BN2	0.98161 0.0001	0.96174 0.0001	0.96770 0.0001	1.00000 0.0000	0.99966 0.0001	0.98839 0.0001	0.99313 0.0001	-0.74194 0.0010	0.89287 0.0001	0.96620 0.0001	0.98199 0.0001	0.97763 0.0001	0.96284 0.0001
E_V	0.99169 0.0001	0.96467 0.0001	0.96646 0.0001	0.99966 0.0001	1.00000 0.0000	0.98677 0.0001	0.99505 0.0001	-0.72595 0.0015	0.90154 0.0001	0.96554 0.0001	0.98361 0.0001	0.97837 0.0001	0.96402 0.0001
E_D	0.98569 0.0001	0.94726 0.0001	0.97479 0.0001	0.99839 0.0001	0.99677 0.0001	1.00000 0.0000	0.98508 0.0001	-0.77866 0.0004	0.86638 0.0001	0.96758 0.0001	0.97191 0.0001	0.96843 0.0001	0.95268 0.0001
E_E	0.99626 0.0001	0.98514 0.0001	0.93839 0.0001	0.99313 0.0001	0.99505 0.0001	0.98508 0.0001	1.00000 0.0000	-0.66169 0.0052	0.92938 0.0001	0.98152 0.0001	0.98537 0.0001	0.98752 0.0001	0.97758 0.0001
E_L	-0.68899 0.0032	-0.56719 0.0219	-0.81158 0.0001	-0.74194 0.0010	-0.72595 0.0015	-0.77866 0.0004	-0.66169 0.0052	1.00000 0.0000	-0.37008 0.1583	-0.77147 0.0005	-0.63300 0.0085	-0.65233 0.0062	-0.61913 0.0105
E_VG	0.92800 0.0001	0.87250 0.0001	0.77857 0.0004	0.89287 0.0001	0.90154 0.0001	0.86638 0.0001	0.92938 0.0001	-0.37008 0.1583	1.00000 0.0000	0.96027 0.0001	0.95288 0.0001	0.93516 0.0001	0.93987 0.0001
E_N	0.97857 0.0001	0.93553 0.0001	0.88599 0.0001	0.99620 0.0001	0.99554 0.0001	0.99758 0.0001	0.98152 0.0001	-0.77147 0.0005	0.96027 0.0001	1.00000 0.0000	0.96237 0.0001	0.95910 0.0001	0.94054 0.0001
E_NE	0.98719 0.0001	0.98606 0.0001	0.90434 0.0001	0.98199 0.0001	0.98361 0.0001	0.97191 0.0001	0.98537 0.0001	-0.63300 0.0085	0.95288 0.0001	0.98237 0.0001	1.00000 0.0000	0.98186 0.0001	0.98588 0.0001
N_N1	0.89134 0.0001	0.98578 0.0001	0.90322 0.0001	0.97763 0.0001	0.97837 0.0001	0.96843 0.0001	0.98752 0.0001	-0.65233 0.0062	0.93516 0.0001	0.95910 0.0001	0.98186 0.0001	1.00000 0.0000	0.95964 0.0001
N_N2	0.86233 0.0001	0.98388 0.0001	0.87783 0.0001	0.96284 0.0001	0.96402 0.0001	0.95268 0.0001	0.97758 0.0001	-0.61913 0.0105	0.93987 0.0001	0.94054 0.0001	0.98588 0.0001	0.95964 0.0001	1.00000 0.0000
N_BN1	0.96590 0.0001	0.96478 0.0001	0.90333 0.0001	0.96077 0.0001	0.96473 0.0001	0.94879 0.0001	0.97484 0.0001	-0.58923 0.0163	0.93858 0.0001	0.94798 0.0001	0.97014 0.0001	0.95890 0.0001	0.94652 0.0001
N_BN2	0.97266 0.0001	0.98036 0.0001	0.88680 0.0001	0.95963 0.0001	0.96406 0.0001	0.94513 0.0001	0.98002 0.0001	-0.56041 0.0239	0.96080 0.0001	0.94151 0.0001	0.98064 0.0001	0.96359 0.0001	0.96611 0.0001
N_V	0.42408 0.1016	0.37593 0.1513	0.51168 0.0428	0.46137 0.0720	0.46003 0.0730	0.46993 0.0663	0.43423 0.0928	-0.44366 0.0852	0.32242 0.2233	0.48216 0.0596	0.40545 0.1192	0.41233 0.1125	0.38319 0.1428
N_D	0.94261 0.0001	0.93874 0.0001	0.89015 0.0001	0.94056 0.0001	0.94443 0.0001	0.92952 0.0001	0.95255 0.0001	-0.58405 0.0175	0.91186 0.0001	0.93006 0.0001	0.94566 0.0001	0.96108 0.0001	0.88690 0.0001

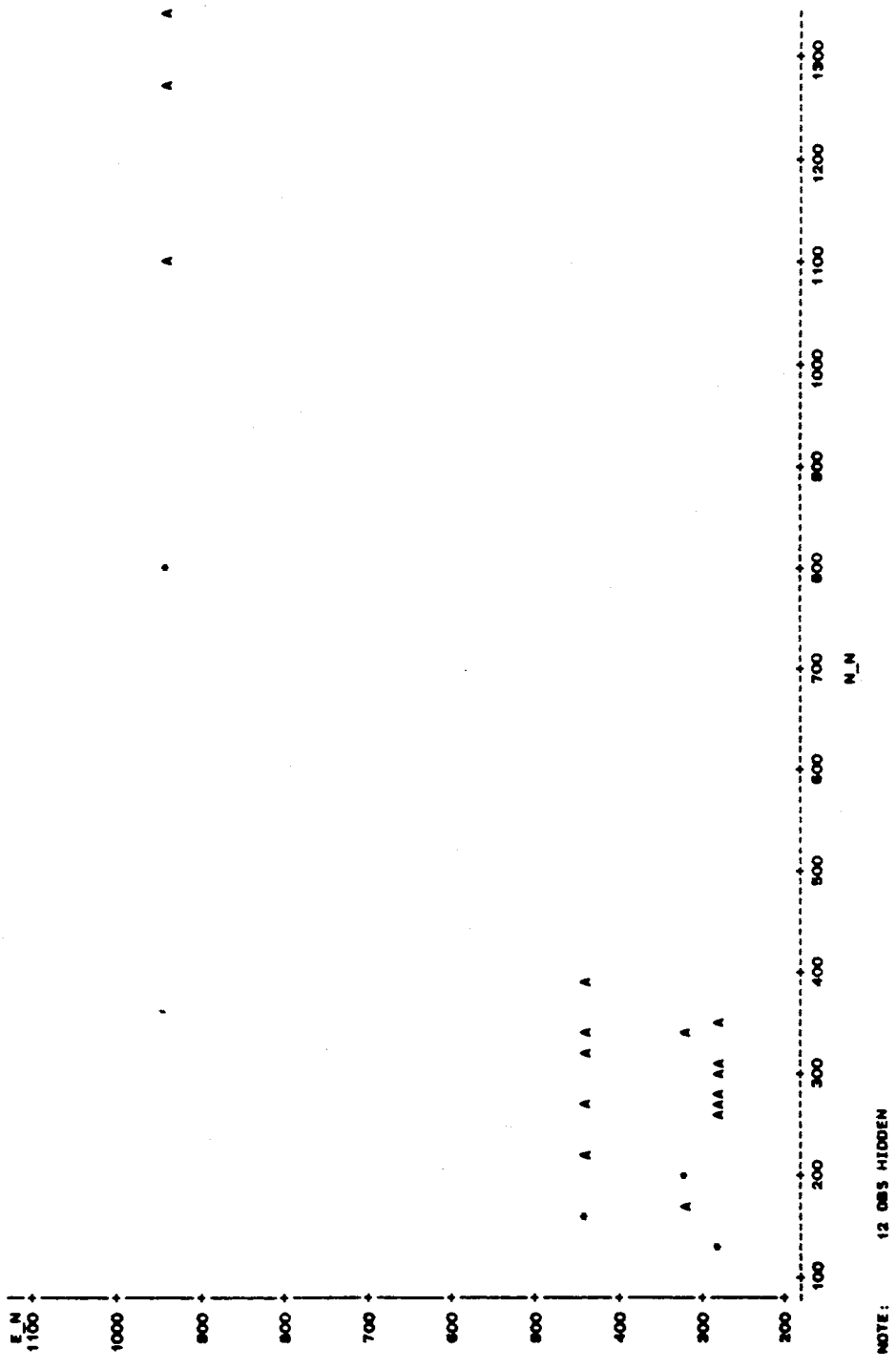
PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 16

	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE	N_N1	N_N2
N_E	0.82713 0.0001	0.82811 0.0001	0.79622 0.0002	0.83070 0.0001	0.83664 0.0001	0.81765 0.0001	0.84544 0.0001	-0.47286 0.0643	0.82314 0.0001	0.82490 0.0001	0.83312 0.0001	0.86001 0.0001	0.74959 0.0008
N_L	0.62481 0.0097	0.61602 0.0111	0.49028 0.0539	0.58840 0.0165	0.58045 0.0184	0.59226 0.0156	0.58679 0.0169	-0.51261 0.0423	0.52672 0.0361	0.55882 0.0244	0.61774 0.0108	0.56736 0.0219	0.71907 0.0017
N_VG	0.93274 0.0001	0.87201 0.0001	0.79526 0.0002	0.90144 0.0001	0.90868 0.0001	0.87651 0.0001	0.84431 0.0001	-0.39440 0.1306	0.99402 0.0001	0.87137 0.0001	0.85532 0.0001	0.93062 0.0001	0.95099 0.0001
N_N	0.97226 0.0001	0.97684 0.0001	0.89420 0.0001	0.96190 0.0001	0.96615 0.0001	0.94830 0.0001	0.98011 0.0001	-0.57207 0.0206	0.95478 0.0001	0.94558 0.0001	0.97891 0.0001	0.96416 0.0001	0.96133 0.0001
N_NE	0.99718 0.0001	0.99606 0.0001	0.90434 0.0001	0.98199 0.0001	0.98361 0.0001	0.97191 0.0001	0.99537 0.0001	-0.63300 0.0085	0.95288 0.0001	0.96237 0.0001	1.00000 0.0000	0.99186 0.0001	0.98588 0.0001

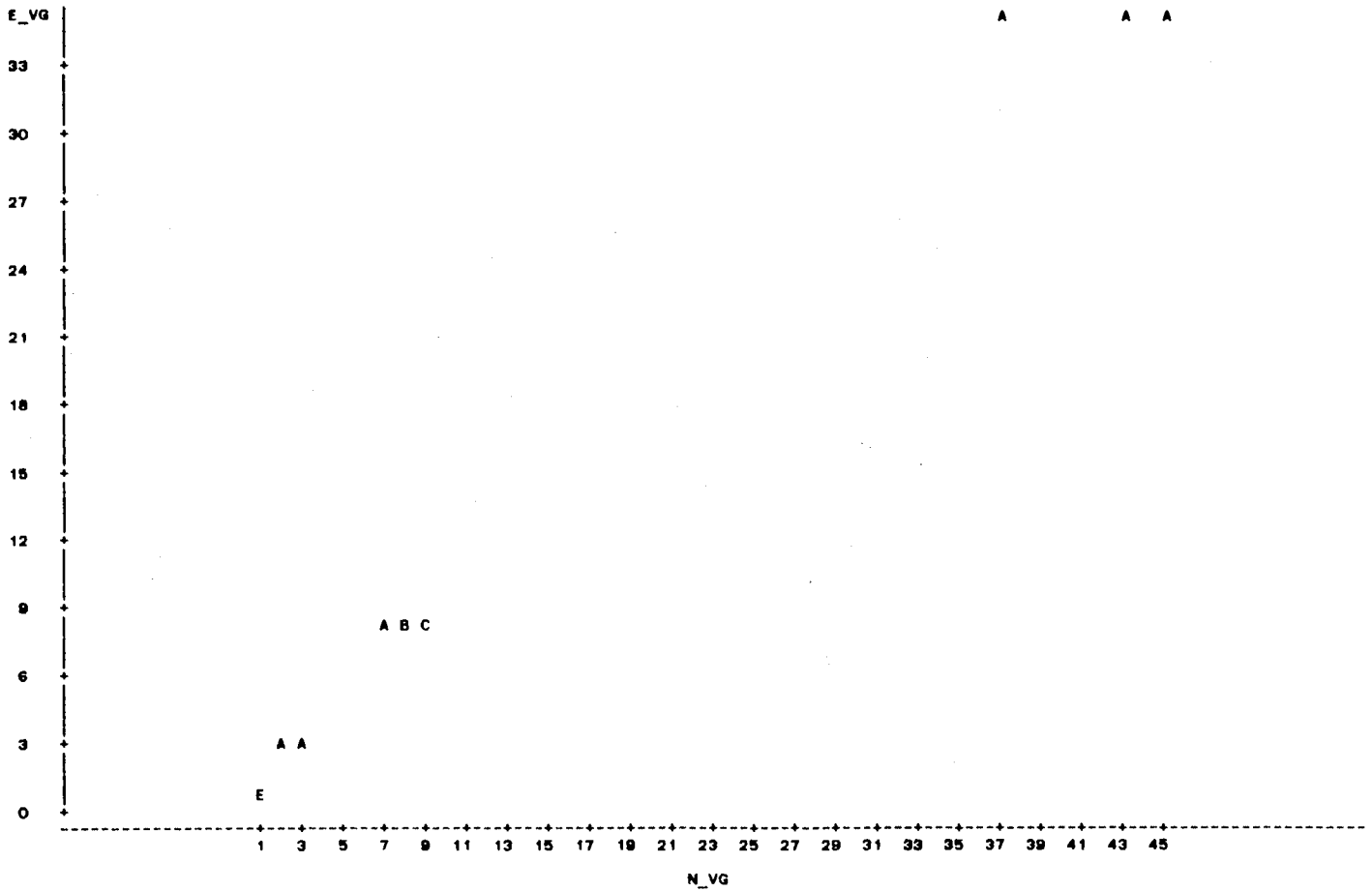
	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
E_N1	0.96590 0.0001	0.97266 0.0001	0.42409 0.1016	0.94261 0.0001	0.82713 0.0001	0.62481 0.0097	0.93274 0.0001	0.97226 0.0001	0.99718 0.0001
E_N2	0.96478 0.0001	0.96036 0.0001	0.37593 0.1513	0.93874 0.0001	0.82811 0.0001	0.61602 0.0111	0.87201 0.0001	0.97684 0.0001	0.99606 0.0001
E_BN1	0.90333 0.0001	0.88680 0.0001	0.51169 0.0428	0.89015 0.0001	0.79622 0.0002	0.49028 0.0539	0.79526 0.0002	0.89420 0.0001	0.90434 0.0001
E_BN2	0.96077 0.0001	0.95963 0.0001	0.46137 0.0720	0.94066 0.0001	0.83070 0.0001	0.58840 0.0165	0.90144 0.0001	0.96190 0.0001	0.98199 0.0001
E_V	0.96473 0.0001	0.96406 0.0001	0.46003 0.0730	0.94443 0.0001	0.83664 0.0001	0.58045 0.0184	0.90868 0.0001	0.96615 0.0001	0.98361 0.0001
E_D	0.94878 0.0001	0.94513 0.0001	0.46993 0.0663	0.92952 0.0001	0.81765 0.0001	0.59226 0.0156	0.87651 0.0001	0.94830 0.0001	0.97191 0.0001
E_E	0.97494 0.0001	0.98002 0.0001	0.43423 0.0928	0.95255 0.0001	0.84544 0.0001	0.58679 0.0169	0.84431 0.0001	0.98011 0.0001	0.99537 0.0001
E_L	-0.58823 0.0163	-0.56041 0.0239	-0.44366 0.0852	-0.58405 0.0175	-0.47286 0.0643	-0.51261 0.0423	-0.39440 0.1306	-0.57207 0.0206	-0.63300 0.0085
E_VG	0.92859 0.0001	0.96090 0.0001	0.32242 0.2233	0.91188 0.0001	0.82314 0.0001	0.52672 0.0361	0.99402 0.0001	0.95478 0.0001	0.95288 0.0001
E_N	0.94798 0.0001	0.94151 0.0001	0.48216 0.0596	0.93006 0.0001	0.82490 0.0001	0.55882 0.0244	0.87137 0.0001	0.94558 0.0001	0.96237 0.0001
E_NE	0.97014 0.0001	0.96064 0.0001	0.40645 0.1192	0.94566 0.0001	0.83312 0.0001	0.61774 0.0108	0.85532 0.0001	0.97891 0.0001	1.00000 0.0000
N_N1	0.95990 0.0001	0.96358 0.0001	0.41233 0.1125	0.96108 0.0001	0.86001 0.0001	0.56736 0.0219	0.93062 0.0001	0.96416 0.0001	0.99186 0.0001

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 16

	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
N_N2	0.94652 0.0001	0.96611 0.0001	0.38319 0.1429	0.88890 0.0001	0.74959 0.0008	0.71907 0.0017	0.95099 0.0001	0.96133 0.0001	0.98588 0.0001
N_BN1	1.00000 0.0000	0.99363 0.0001	0.44855 0.0806	0.97429 0.0001	0.90652 0.0001	0.48845 0.0549	0.95312 0.0001	0.99727 0.0001	0.97014 0.0001
N_BN2	0.99363 0.0001	1.00000 0.0000	0.41392 0.1110	0.96130 0.0001	0.88302 0.0001	0.53588 0.0324	0.97341 0.0001	0.99922 0.0001	0.98064 0.0001
N_V	0.44855 0.0806	0.41392 0.1110	1.00000 0.0000	0.43198 0.0947	0.39117 0.1341	0.12752 0.6379	0.33893 0.1991	0.42672 0.0993	0.40545 0.1192
N_D	0.97429 0.0001	0.96130 0.0001	0.43198 0.0947	1.00000 0.0000	0.96388 0.0001	0.35576 0.1763	0.91214 0.0001	0.96706 0.0001	0.94566 0.0001
N_E	0.90652 0.0001	0.88302 0.0001	0.39117 0.1341	0.96388 0.0001	1.00000 0.0000	0.14806 0.5842	0.82587 0.0001	0.89160 0.0001	0.83312 0.0001
N_L	0.48845 0.0549	0.53588 0.0324	0.12752 0.6379	0.35576 0.1763	0.14806 0.5842	1.00000 0.0000	0.55405 0.0260	0.52156 0.0383	0.61774 0.0108
N_VG	0.95312 0.0001	0.97341 0.0001	0.33893 0.1991	0.91214 0.0001	0.82587 0.0001	0.55405 0.0260	1.00000 0.0000	0.96801 0.0001	0.95532 0.0001
N_N	0.99727 0.0001	0.99922 0.0001	0.42672 0.0993	0.96706 0.0001	0.89160 0.0001	0.52156 0.0383	0.96801 0.0001	1.00000 0.0000	0.97891 0.0001
N_NE	0.97014 0.0001	0.98064 0.0001	0.40545 0.1192	0.94566 0.0001	0.83312 0.0001	0.61774 0.0108	0.95532 0.0001	0.97891 0.0001	1.00000 0.0000


SAS
12:30 WEDNESDAY, JUNE 7, 1988 7
PLOT OF E_N*N_N
PLOT OF E_N*N_ME
LEGEND: A = 1 OBS, B = 2 OBS, ETC.
SYMBOL USED IS *
E_N
N_N
NOTE: 12 OBS HIDDEN

PLOT OF E_VG*N_VG LEGEND: A = 1 OBS, B = 2 OBS, ETC.



The following tables and plots are for the expert C++ programs and novice C++ programs analysis and comparison. The novice metrics are extrapolated with N. The variables are represented as

g_m.

Where g could be E for expert; or

N for novice;

and

m is the selected metric. (see nomenclature)

OBS	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE
1	70	39	376	633	4734	568	2688912	0.015	24	1009	440
2	70	39	376	633	4734	568	2688912	0.015	24	1009	440
3	70	39	376	633	4734	568	2688912	0.015	24	1009	440
4	70	39	376	633	4734	564	2688912	0.015	24	1009	440
5	70	39	376	633	4734	564	2688912	0.015	24	1009	440
6	30	25	156	198	1426	119	169481	0.101	1	356	183
7	17	23	101	170	1000	63	62806	0.253	8	271	120
8	17	23	101	170	1000	63	62806	0.253	8	271	120
9	17	23	101	170	1000	63	62806	0.253	8	271	120
10	17	23	101	170	1000	63	62806	0.253	8	271	120

OBS	N_N1	N_N2	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
1	60	47	199	295	2038	250	248665	0.1230	12	494	440
2	77	39	170	275	1969	275	391851	0.0550	12	444	440
3	75	47	171	295	1991	232	272268	0.1020	8	466	440
4	78	56	182	319	1989	225	176579	0.2530	23	501	440
5	72	62	200	326	2100	190	160819	0.3460	20	527	440
6	30	30	177	226	1507	113	119047	0.2430	1	403	183
7	18	22	29	72	379	30	12036	0.3750	4	101	120
8	19	16	54	94	607	58	59344	0.0710	4	148	120
9	17	21	38	58	365	23	9939	0.4890	2	96	120
10	20	14	69	129	823	94	144442	0.0289	5	198	120

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
E_N1	10	44.800000	26.844408	448.00000	17.00000000	70.000000
E_N2	10	31.200000	8.243516	312.00000	23.00000000	39.000000
E_BN1	10	244.200000	139.964916	2442.00000	101.00000000	376.000000
E_BN2	10	404.300000	241.215464	4043.00000	170.00000000	632.000000
E_V	10	2809.600000	1927.275982	28096.00000	1000.00000000	4734.000000
E_D	10	320.300000	259.953008	3203.00000	63.00000000	568.000000
E_E	10	1386526.500000	1373203.211902	13865265.00000	62806.00000000	2688912.000000
E_L	10	1.188000000E-01	1.184284125E-01	1.18800	1.500000000E-02	2.530000000E-01
E_VG	10	15.300000	8.405091	153.00000	1.000000000E+00	24.000000
E_N	10	648.500000	380.844456	6485.00000	271.00000000	1008.000000
E_NE	10	286.300000	163.099186	2863.00000	120.00000000	440.000000
N_N1	10	48.600000	29.583028	486.00000	17.00000000	80.000000
N_N2	10	35.400000	17.208842	354.00000	14.00000000	62.000000
N_BN1	10	128.900000	71.495843	1289.00000	28.00000000	200.000000
N_BN2	10	208.800000	108.740721	2088.00000	58.00000000	326.000000
N_V	10	1376.800000	745.245493	13768.00000	365.00000000	2100.000000
N_D	10	149.000000	96.068263	1490.00000	23.00000000	275.000000
N_E	10	159508.000000	120626.258505	1595080.00000	8930.00000000	391851.000000
N_L	10	2.083900000E-01	1.572608198E-01	2.08390	2.690000000E-02	4.890000000E-01
N_VG	10	9.100000	7.563803	91.00000	1.000000000E+00	23.000000
N_N	10	337.900000	178.165845	3378.00000	96.00000000	527.000000
N_NE	10	286.300000	163.099186	2863.00000	120.00000000	440.000000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE	N_N1	N_N2
E_N1	1.00000 0.0000	0.99737 0.0001	0.99973 0.0001	0.99393 0.0001	0.99689 0.0001	0.99674 0.0001	0.99260 0.0001	-0.96946 0.0001	0.93281 0.0001	0.99684 0.0001	0.99957 0.0001	0.99691 0.0001	0.92659 0.0001
E_N2	0.99737 0.0001	1.00000 0.0000	0.99879 0.0001	0.99929 0.0001	0.99999 0.0001	0.99995 0.0001	0.99879 0.0001	-0.94815 0.0001	0.95646 0.0001	0.99998 0.0001	0.99907 0.0001	0.99635 0.0001	0.91905 0.0002
E_BN1	0.99973 0.0001	0.99879 0.0001	1.00000 0.0000	0.99621 0.0001	0.99845 0.0001	0.99833 0.0001	0.99515 0.0001	-0.96352 0.0001	0.94081 0.0001	0.99848 0.0001	0.99998 0.0001	0.99730 0.0001	0.92471 0.0001
E_BN2	0.99393 0.0001	0.99929 0.0001	0.99621 0.0001	1.00000 0.0000	0.99851 0.0001	0.99854 0.0001	0.99993 0.0001	-0.93658 0.0001	0.96681 0.0001	0.99949 0.0001	0.99674 0.0001	0.99389 0.0001	0.91320 0.0002
E_V	0.99689 0.0001	0.99999 0.0001	0.99845 0.0001	0.99851 0.0001	1.00000 0.0000	0.99998 0.0001	0.99908 0.0001	-0.94710 0.0001	0.95833 0.0001	1.00000 0.0001	0.99976 0.0001	0.99704 0.0001	0.91814 0.0002
E_D	0.99674 0.0001	0.99995 0.0001	0.99833 0.0001	0.99854 0.0001	0.99998 0.0001	1.00000 0.0000	0.99914 0.0001	-0.94654 0.0001	0.95879 0.0001	0.99999 0.0001	0.99968 0.0001	0.99711 0.0001	0.91814 0.0002
E_E	0.99260 0.0001	0.99879 0.0001	0.99515 0.0001	0.99993 0.0001	0.99908 0.0001	0.99914 0.0001	1.00000 0.0000	-0.93250 0.0001	0.96967 0.0001	0.99906 0.0001	0.99575 0.0001	0.99298 0.0001	0.91117 0.0002
E_L	-0.96946 0.0001	-0.94815 0.0001	-0.96352 0.0001	-0.93658 0.0001	-0.94710 0.0001	-0.94654 0.0001	-0.93250 0.0001	1.00000 0.0000	-0.91584 0.0040	-0.94730 0.0001	-0.96181 0.0001	-0.95952 0.0001	-0.91560 0.0002
E_VG	0.93281 0.0001	0.95646 0.0001	0.94081 0.0001	0.96681 0.0001	0.95833 0.0001	0.95879 0.0001	0.96967 0.0001	-0.91584 0.0040	1.00000 0.0000	0.95814 0.0001	0.94303 0.0001	0.94015 0.0001	0.93891 0.0024
E_N	0.99684 0.0001	0.99998 0.0001	0.99848 0.0001	0.99949 0.0001	1.00000 0.0001	0.99999 0.0001	0.99906 0.0001	-0.94730 0.0001	0.95814 0.0001	1.00000 0.0000	0.99981 0.0001	0.99708 0.0001	0.91824 0.0002
E_NE	0.99957 0.0001	0.99907 0.0001	0.99998 0.0001	0.99674 0.0001	0.99976 0.0001	0.99968 0.0001	0.99575 0.0001	-0.96181 0.0001	0.94303 0.0001	0.99981 0.0001	1.00000 0.0000	0.99732 0.0001	0.92411 0.0001
N_N1	0.99691 0.0001	0.99635 0.0001	0.99730 0.0001	0.99389 0.0001	0.99704 0.0001	0.99711 0.0001	0.99298 0.0001	-0.95952 0.0001	0.94015 0.0001	0.99708 0.0001	0.99732 0.0001	1.00000 0.0000	0.90648 0.0003
N_N2	0.92659 0.0001	0.91905 0.0002	0.92471 0.0001	0.91320 0.0002	0.91814 0.0002	0.91814 0.0002	0.91117 0.0002	-0.91560 0.0002	0.93891 0.0024	0.91824 0.0002	0.92411 0.0001	0.90648 0.0003	1.00000 0.0000
N_BN1	0.88766 0.0006	0.85518 0.0016	0.87777 0.0008	0.83646 0.0026	0.85207 0.0017	0.85083 0.0018	0.83055 0.0029	-0.96263 0.0001	0.67803 0.0312	0.85236 0.0017	0.87501 0.0009	0.87571 0.0008	0.96325 0.0013
N_BN2	0.94755 0.0001	0.92743 0.0001	0.94166 0.0001	0.91501 0.0002	0.92540 0.0001	0.92421 0.0001	0.91098 0.0002	-0.97631 0.0001	0.79617 0.0059	0.92561 0.0001	0.93997 0.0001	0.93799 0.0001	0.91885 0.0002
N_V	0.95224 0.0001	0.93159 0.0001	0.94618 0.0001	0.91889 0.0002	0.92951 0.0001	0.92883 0.0001	0.91476 0.0002	-0.96462 0.0001	0.78796 0.0067	0.92972 0.0001	0.94444 0.0001	0.94459 0.0001	0.89077 0.0008
N_D	0.95489 0.0001	0.94844 0.0001	0.95337 0.0001	0.94311 0.0001	0.94763 0.0001	0.94835 0.0001	0.94122 0.0001	-0.93905 0.0001	0.87115 0.0010	0.94771 0.0001	0.95267 0.0001	0.96544 0.0001	0.78252 0.0063

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

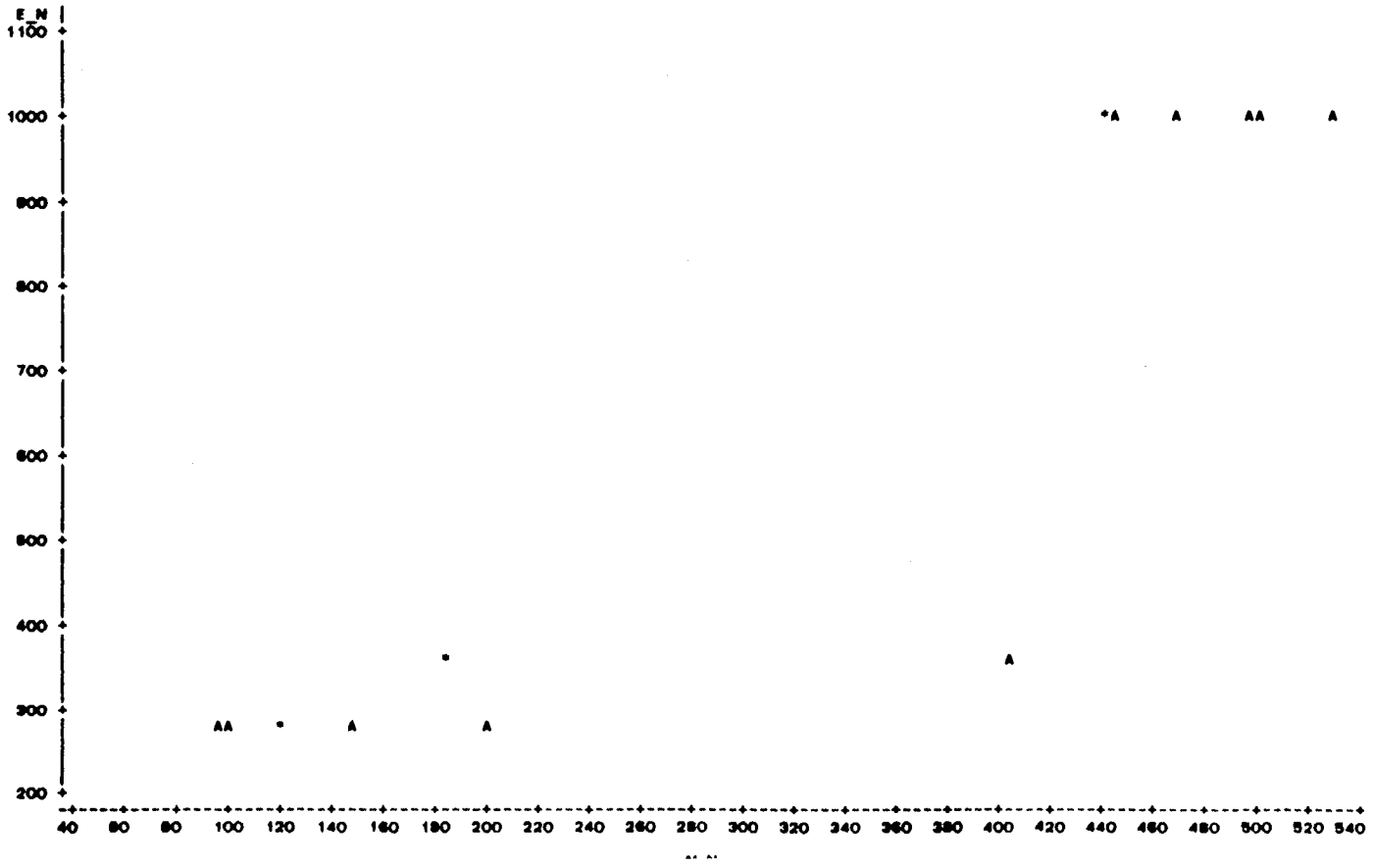
	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE	N_N1	N_N2
N_E	0.80530 0.0049	0.80037 0.0054	0.80418 0.0050	0.79613 0.0059	0.79973 0.0055	0.80185 0.0053	0.79462 0.0060	-0.79024 0.0065	0.73718 0.0150	0.79979 0.0055	0.80380 0.0051	0.81881 0.0038	0.84509 0.1032
N_L	-0.21547 0.5499	-0.21753 0.5480	-0.21825 0.5485	-0.21815 0.5449	-0.21765 0.5458	-0.22037 0.5407	-0.21828 0.5446	0.19999 0.5796	-0.21406 0.5526	-0.21764 0.5458	-0.21644 0.5481	-0.25475 0.4775	0.07767 0.8311
N_VG	0.79796 0.0057	0.81223 0.0043	0.80298 0.0052	0.81788 0.0038	0.81329 0.0042	0.81061 0.0044	0.81949 0.0037	-0.71817 0.0193	0.82578 0.0032	0.81319 0.0042	0.80427 0.0050	0.79818 0.0056	0.96111 0.0014
N_N	0.82931 0.0001	0.90414 0.0003	0.82179 0.0001	0.88913 0.0006	0.90167 0.0004	0.90048 0.0004	0.88433 0.0007	-0.97790 0.0001	0.75378 0.0118	0.90192 0.0004	0.91966 0.0002	0.91864 0.0002	0.90298 0.0003
N_NE	0.99857 0.0001	0.99907 0.0001	0.99998 0.0001	0.99674 0.0001	0.99878 0.0001	0.99888 0.0001	0.99575 0.0001	-0.96181 0.0001	0.94303 0.0001	0.99881 0.0001	1.00000 0.0001	0.99732 0.0001	0.92411 0.0001

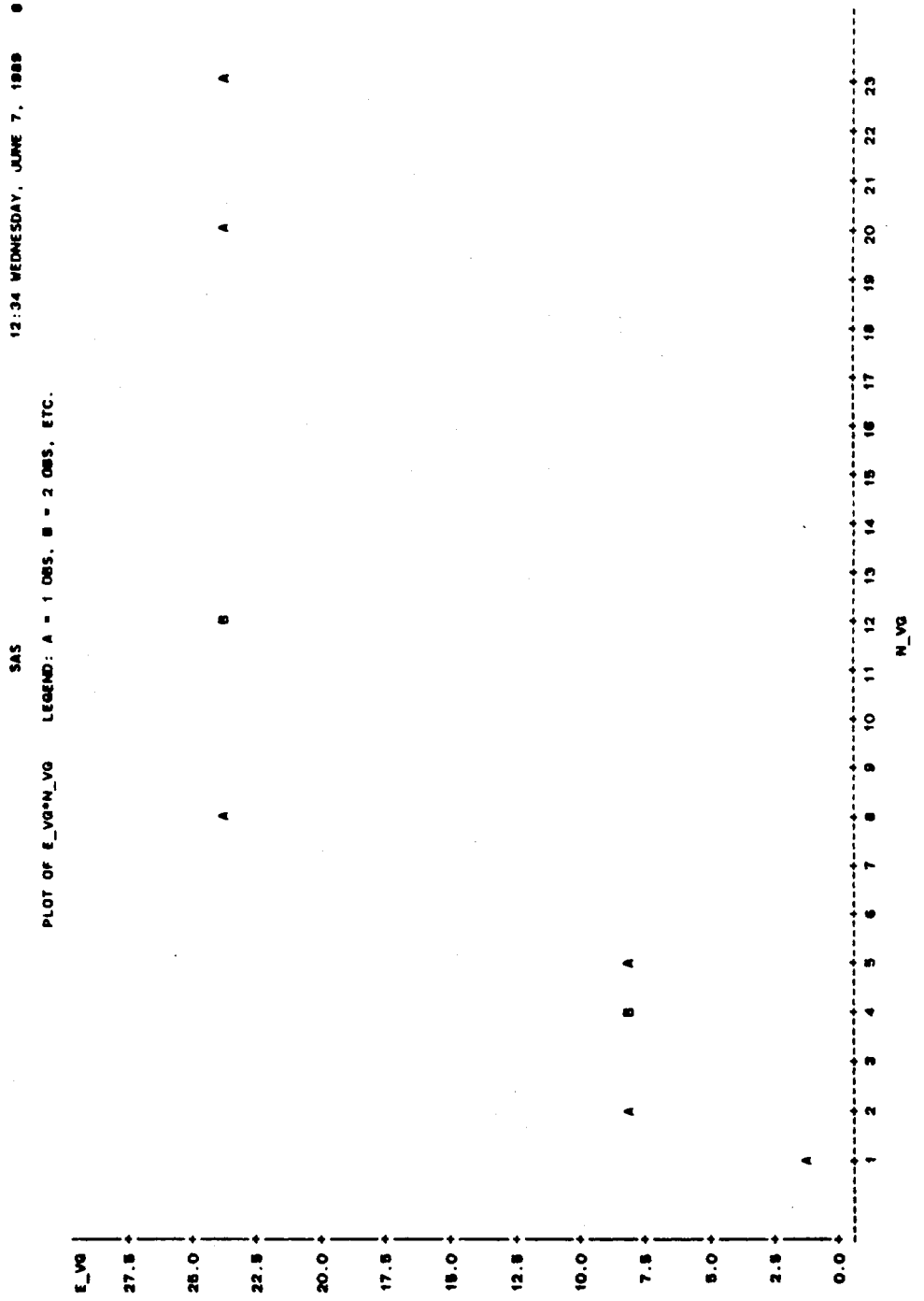
	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
E_N1	0.88766 0.0008	0.84755 0.0001	0.85224 0.0001	0.95488 0.0001	0.80530 0.0049	-0.21547 0.5499	0.79796 0.0057	0.82931 0.0001	0.99857 0.0001
E_N2	0.85518 0.0018	0.92743 0.0001	0.93159 0.0001	0.94844 0.0001	0.80037 0.0054	-0.21753 0.5480	0.81223 0.0043	0.90414 0.0003	0.99907 0.0001
E_BN1	0.87777 0.0008	0.84186 0.0001	0.84618 0.0001	0.95337 0.0001	0.80418 0.0050	-0.21825 0.5485	0.80298 0.0052	0.82179 0.0001	0.99998 0.0001
E_BN2	0.83646 0.0026	0.91501 0.0002	0.91888 0.0002	0.94311 0.0001	0.79613 0.0059	-0.21815 0.5449	0.81788 0.0038	0.88913 0.0006	0.99674 0.0001
E_V	0.85207 0.0017	0.92540 0.0001	0.92951 0.0001	0.94763 0.0001	0.79973 0.0055	-0.21765 0.5458	0.81329 0.0042	0.90167 0.0004	0.99878 0.0001
E_D	0.85093 0.0018	0.92421 0.0001	0.92883 0.0001	0.94835 0.0001	0.80185 0.0053	-0.22037 0.5407	0.81061 0.0044	0.90048 0.0004	0.99888 0.0001
E_E	0.83055 0.0029	0.91088 0.0002	0.91476 0.0002	0.94122 0.0001	0.79462 0.0060	-0.21828 0.5446	0.81949 0.0037	0.88433 0.0007	0.99575 0.0001
E_L	-0.96263 0.0001	-0.97831 0.0001	-0.98482 0.0001	-0.93806 0.0001	-0.79024 0.0065	0.19999 0.5796	-0.71817 0.0193	-0.97790 0.0001	-0.96181 0.0001
E_VG	0.87603 0.0312	0.79617 0.0059	0.79796 0.0057	0.87115 0.0010	0.73718 0.0150	-0.21406 0.5526	0.82578 0.0032	0.75378 0.0118	0.94303 0.0001
E_N	0.85238 0.0017	0.82561 0.0001	0.82972 0.0001	0.84771 0.0001	0.79979 0.0055	-0.21764 0.5458	0.81319 0.0042	0.90192 0.0004	0.99881 0.0001
E_NE	0.87501 0.0009	0.83897 0.0001	0.84444 0.0001	0.95287 0.0001	0.80380 0.0051	-0.21644 0.5481	0.80427 0.0050	0.91966 0.0002	1.00000 0.0001
N_N1	0.87571	0.83799	0.84459	0.96544	0.81881	-0.25475	0.79818	0.91864	0.99732

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
N_N2	0.88325 0.0013	0.91885 0.0002	0.89077 0.0005	0.78252 0.0063	0.54509 0.1032	0.07767 0.8311	0.86111 0.0014	0.90288 0.0003	0.92411 0.0001
N_BN1	1.00000 0.0000	0.97409 0.0001	0.97849 0.0001	0.88041 0.0008	0.72586 0.0175	-0.23660 0.5105	0.85381 0.0403	0.99050 0.0001	0.87501 0.0009
N_BN2	0.97409 0.0001	1.00000 0.0000	0.99438 0.0001	0.92406 0.0001	0.75870 0.0110	-0.25825 0.4695	0.76878 0.0092	0.99592 0.0001	0.93997 0.0001
N_V	0.97849 0.0001	0.99438 0.0001	1.00000 0.0000	0.94763 0.0001	0.81073 0.0044	-0.30831 0.3861	0.72471 0.0177	0.99409 0.0001	0.94444 0.0001
N_D	0.88041 0.0008	0.92406 0.0001	0.94763 0.0001	1.00000 0.0000	0.92803 0.0001	-0.45374 0.1878	0.69620 0.0253	0.91162 0.0002	0.96287 0.0001
N_E	0.72586 0.0175	0.75870 0.0110	0.81073 0.0044	0.92803 0.0001	1.00000 0.0000	-0.61934 0.0562	0.45876 0.1813	0.74894 0.0127	0.80380 0.0051
N_L	-0.23660 0.5105	-0.25825 0.4695	-0.30831 0.3861	-0.45374 0.1878	-0.61934 0.0562	1.00000 0.0000	-0.01242 0.9728	-0.25061 0.4849	-0.21644 0.5451
N_VG	0.85381 0.0403	0.76878 0.0092	0.72471 0.0177	0.69620 0.0253	0.45876 0.1813	-0.01242 0.9728	1.00000 0.0000	0.72875 0.0168	0.90427 0.0050
N_N	0.99050 0.0001	0.99592 0.0001	0.99409 0.0001	0.91162 0.0002	0.74894 0.0127	-0.25061 0.4849	0.72875 0.0168	1.00000 0.0000	0.91966 0.0002
N_NE	0.87501 0.0009	0.93997 0.0001	0.94444 0.0001	0.96287 0.0001	0.80380 0.0051	-0.21644 0.5451	0.90427 0.0050	0.91966 0.0002	1.00000 0.0000

PLOT OF E_N*N_N LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF E_N*N_NE SYMBOL USED IS *




SAS
12:34 WEDNESDAY, JUNE 7, 1989
PLOT OF E_VG*N_VG LEGEND: A = 1 OBS, B = 2 OBS, ETC.
E_VG
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
N_VG

The following tables and plots are for the expert C++ programs and novice C++ programs analysis and comparison. The novice metrics are extrapolated with Ne. The variables are represented as

g_m.

Where g could be E for expert; or

N for novice;

and

m is the selected metric. (see nomenclature)

OBS	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VQ	E_N	E_NE
1	70	39	376	633	4734	568	2688912	0.015	24	1009	440
2	70	39	376	633	4734	568	2688912	0.015	24	1009	440
3	70	39	376	633	4734	568	2688912	0.015	24	1009	440
4	70	39	376	633	4734	564	2688912	0.015	24	1009	440
5	70	39	376	633	4734	564	2688912	0.015	24	1009	440
6	30	25	156	198	1426	119	169481	0.101	1	356	183
7	17	23	101	170	1000	63	62806	0.253	8	271	120
8	17	23	101	170	1000	63	62806	0.253	6	271	120
9	17	23	101	170	1000	63	62806	0.253	8	271	120
10	17	23	101	170	1000	63	62806	0.253	8	271	120

OBS	N_N1	N_N2	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
1	163	96	406	603	4166	511	508402	0.251	25	1009	900
2	175	88	385	624	4472	624	889669	0.125	28	1009	1000
3	161	103	371	638	4315	503	589942	0.220	18	1009	954
4	158	112	367	642	4005	454	355627	0.510	46	1009	687
5	137	118	384	625	4022	365	308180	0.663	38	1009	843
6	27	27	157	199	1331	100	105119	0.215	1	356	162
7	48	58	78	193	1014	80	32214	1.004	10	271	321
8	35	29	99	172	1110	105	108490	0.129	8	271	219
9	48	60	106	165	1031	65	28058	1.380	5	271	339
10	28	19	94	177	1127	129	197669	0.037	7	271	164

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
E_N1	10	44.800000	26.844408	448.00000	17.00000000	70.000000
E_N2	10	31.200000	8.243516	312.00000	23.00000000	39.000000
E_SN1	10	244.200000	139.964916	2442.00000	101.00000000	376.000000
E_SN2	10	404.300000	241.218464	4043.00000	170.00000000	633.000000
E_V	10	2809.600000	1927.275882	28096.00000	1000.00000000	4734.000000
E_D	10	320.300000	259.853009	3203.00000	83.00000000	848.000000
E_E	10	1366526.800000	1373203.211902	13665268.00000	82806.00000000	2688812.000000
E_L	10	1.188000000E-01	1.184284125E-01	1.18800	1.500000000E-02	2.830000000E-01
E_VG	10	15.300000	9.408081	153.00000	1.000000000E+00	24.000000
E_N	10	648.900000	380.844456	6485.00000	271.00000000	1008.000000
E_NE	10	286.300000	183.099186	2863.00000	120.00000000	440.000000
N_N1	10	98.000000	85.110183	980.00000	27.00000000	178.000000
N_N2	10	71.000000	37.300681	710.00000	19.00000000	118.000000
N_SN1	10	244.700000	147.072204	2447.00000	78.00000000	406.000000
N_SN2	10	403.600000	235.065927	4036.00000	185.00000000	642.000000
N_V	10	2659.300000	1627.412847	26593.00000	1014.00000000	4472.000000
N_D	10	293.600000	218.358942	2936.00000	65.00000000	824.000000
N_E	10	312339.000000	280403.384792	3123390.00000	28056.00000000	889669.000000
N_L	10	4.534000000E-01	4.406598092E-01	4.53400	3.700000000E-02	1.360000
N_VG	10	18.600000	15.174540	186.00000	1.000000000E+00	48.000000
N_N	10	648.900000	380.844456	6485.00000	271.00000000	1008.000000
N_NE	10	578.800000	362.911820	5788.00000	162.00000000	1000.000000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE	N_N1	N_N2
E_N1	1.00000 0.0000	0.99737 0.0001	0.99973 0.0001	0.99393 0.0001	0.99689 0.0001	0.99674 0.0001	0.99260 0.0001	-0.96946 0.0001	0.93281 0.0001	0.99694 0.0001	0.99957 0.0001	0.96557 0.0001	0.88928 0.0006
E_N2	0.99737 0.0001	1.00000 0.0000	0.99878 0.0001	0.99929 0.0001	0.99998 0.0001	0.99995 0.0001	0.99879 0.0001	-0.94915 0.0001	0.95646 0.0001	0.99998 0.0001	0.99907 0.0001	0.97751 0.0001	0.90482 0.0003
E_BN1	0.99973 0.0001	0.99878 0.0001	1.00000 0.0000	0.99621 0.0001	0.99845 0.0001	0.99833 0.0001	0.99515 0.0001	-0.96352 0.0001	0.94091 0.0001	0.99848 0.0001	0.99998 0.0001	0.96994 0.0001	0.89476 0.0005
E_BN2	0.99393 0.0001	0.99929 0.0001	0.99621 0.0001	1.00000 0.0000	0.99951 0.0001	0.99954 0.0001	0.99993 0.0001	-0.93658 0.0001	0.96681 0.0001	0.99949 0.0001	0.99674 0.0001	0.98170 0.0001	0.91105 0.0002
E_V	0.99689 0.0001	0.99998 0.0001	0.99845 0.0001	0.99951 0.0001	1.00000 0.0000	0.99998 0.0001	0.99908 0.0001	-0.94710 0.0001	0.95833 0.0001	1.00000 0.0001	0.99878 0.0001	0.97833 0.0001	0.90598 0.0003
E_D	0.99674 0.0001	0.99995 0.0001	0.99833 0.0001	0.99954 0.0001	0.99998 0.0001	1.00000 0.0000	0.99914 0.0001	-0.94654 0.0001	0.95878 0.0001	0.99998 0.0001	0.99868 0.0001	0.97911 0.0001	0.90519 0.0003
E_E	0.99260 0.0001	0.99879 0.0001	0.99515 0.0001	0.99993 0.0001	0.99908 0.0001	0.99914 0.0001	1.00000 0.0000	-0.93250 0.0001	0.96967 0.0001	0.99906 0.0001	0.99575 0.0001	0.98270 0.0001	0.91268 0.0002
E_L	-0.96946 0.0001	-0.94915 0.0001	-0.96352 0.0001	-0.93658 0.0001	-0.94710 0.0001	-0.94654 0.0001	-0.93250 0.0001	1.00000 0.0000	-0.81594 0.0040	-0.94730 0.0001	-0.96181 0.0001	-0.98706 0.0006	-0.80157 0.0053
E_VG	0.93281 0.0001	0.95646 0.0001	0.94091 0.0001	0.96681 0.0001	0.95833 0.0001	0.95878 0.0001	0.96967 0.0001	-0.81594 0.0040	1.00000 0.0000	0.95814 0.0001	0.94303 0.0001	0.97273 0.0001	0.91850 0.0002
E_N	0.99694 0.0001	0.99998 0.0001	0.99848 0.0001	0.99949 0.0001	1.00000 0.0001	0.99998 0.0001	0.99906 0.0001	-0.94730 0.0001	0.95814 0.0001	1.00000 0.0000	0.99881 0.0001	0.97825 0.0001	0.90588 0.0003
E_NE	0.99957 0.0001	0.99907 0.0001	0.99998 0.0001	0.99674 0.0001	0.99878 0.0001	0.99868 0.0001	0.99575 0.0001	-0.96181 0.0001	0.94303 0.0001	0.99881 0.0001	1.00000 0.0000	0.97104 0.0001	0.89617 0.0004
N_N1	0.96557 0.0001	0.97751 0.0001	0.96994 0.0001	0.98170 0.0001	0.97833 0.0001	0.97911 0.0001	0.98270 0.0001	-0.98706 0.0006	0.97273 0.0001	0.97825 0.0001	0.97104 0.0001	1.00000 0.0000	0.90828 0.0003
N_N2	0.88928 0.0006	0.90482 0.0003	0.89476 0.0005	0.91105 0.0002	0.90598 0.0003	0.90518 0.0003	0.91268 0.0002	-0.80157 0.0053	0.91850 0.0002	0.90588 0.0003	0.89617 0.0004	0.90828 0.0003	1.00000 0.0000
N_BN1	0.99636 0.0001	0.99487 0.0001	0.99649 0.0001	0.99216 0.0001	0.99459 0.0001	0.99463 0.0001	0.99103 0.0001	-0.96181 0.0001	0.93549 0.0001	0.99463 0.0001	0.99643 0.0001	0.96585 0.0001	0.88162 0.0007
N_BN2	0.99185 0.0001	0.99766 0.0001	0.99428 0.0001	0.99861 0.0001	0.99792 0.0001	0.99787 0.0001	0.99862 0.0001	-0.93306 0.0001	0.96708 0.0001	0.99790 0.0001	0.99484 0.0001	0.98066 0.0001	0.91609 0.0002
N_V	0.99180 0.0001	0.99618 0.0001	0.99377 0.0001	0.99639 0.0001	0.99632 0.0001	0.99671 0.0001	0.99618 0.0001	-0.93784 0.0001	0.95993 0.0001	0.99631 0.0001	0.99421 0.0001	0.98418 0.0001	0.88441 0.0007
N_D	0.94587 0.0001	0.95296 0.0001	0.94885 0.0001	0.95452 0.0001	0.95324 0.0001	0.95460 0.0001	0.95475 0.0001	-0.88491 0.0007	0.92945 0.0001	0.95321 0.0001	0.94832 0.0001	0.97245 0.0001	0.79068 0.0064

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

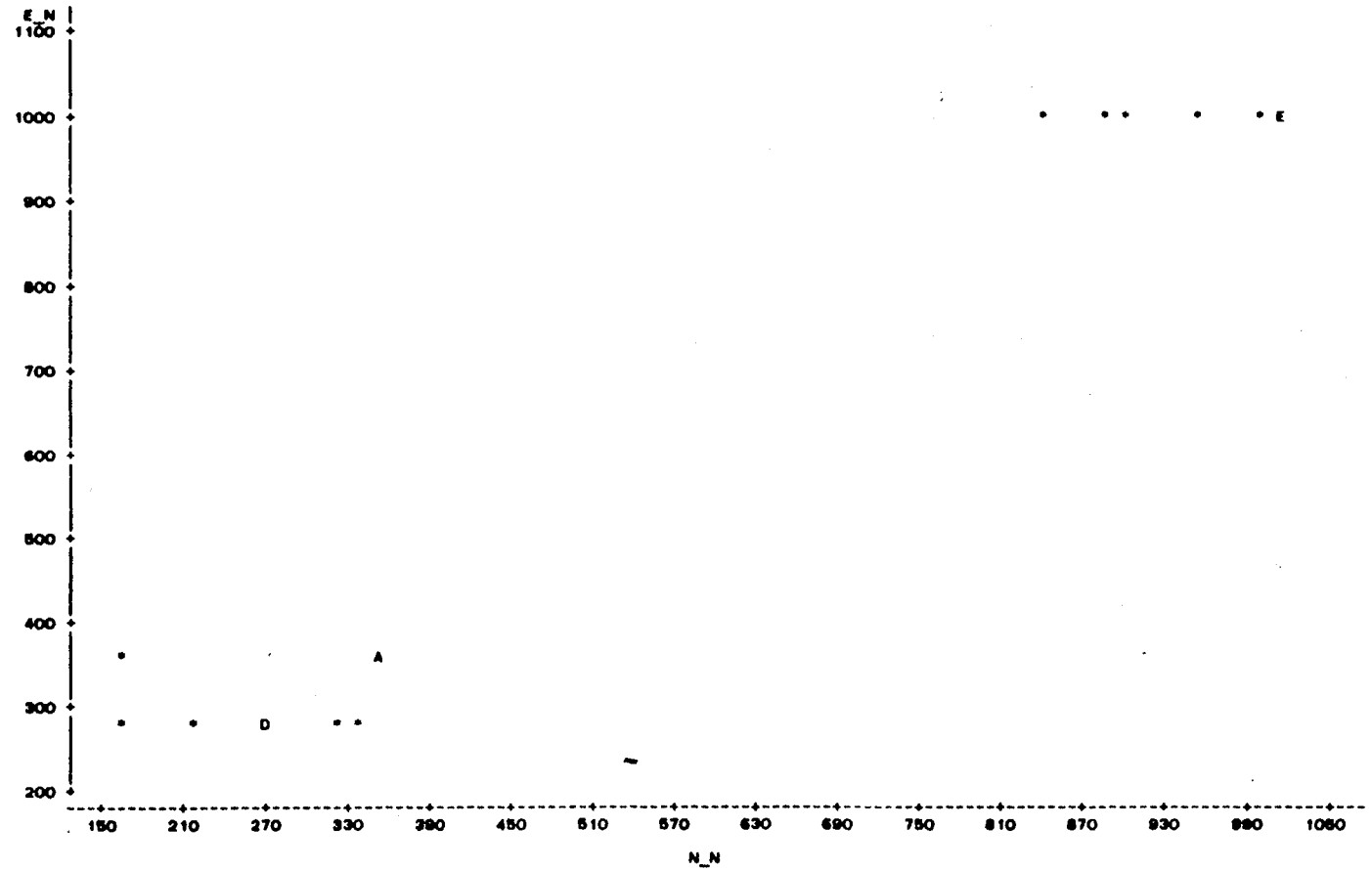
	E_N1	E_N2	E_BN1	E_BN2	E_V	E_D	E_E	E_L	E_VG	E_N	E_NE	N_N1	N_N2
N_E	0.81310 0.0042	0.81850 0.0038	0.81529 0.0040	0.81962 0.0037	0.81878 0.0038	0.82125 0.0036	0.81872 0.0037	-0.76273 0.0103	0.79599 0.0058	0.81875 0.0038	0.81581 0.0040	0.85766 0.0015	0.59065 0.0723
N_L	-0.27703 0.4384	-0.25830 0.4712	-0.27120 0.4485	-0.24800 0.4897	-0.25657 0.4743	-0.25792 0.4719	-0.24481 0.4954	0.32950 0.3525	-0.16868 0.6408	-0.25674 0.4740	-0.26959 0.4513	-0.18947 0.6001	0.15194 0.6752
N_VG	0.83390 0.0027	0.84987 0.0018	0.83948 0.0024	0.85642 0.0016	0.85107 0.0018	0.84889 0.0019	0.85817 0.0015	-0.74693 0.0130	0.86822 0.0011	0.85095 0.0018	0.84092 0.0023	0.82567 0.0033	0.86766 0.0011
N_N	0.99694 0.0001	0.99998 0.0001	0.99848 0.0001	0.99949 0.0001	1.00000 0.0001	0.99998 0.0001	0.99906 0.0001	-0.84730 0.0001	0.95814 0.0001	1.00000 0.0000	0.99881 0.0001	0.97825 0.0001	0.90586 0.0003
N_NE	0.95945 0.0001	0.97301 0.0001	0.96433 0.0001	0.97805 0.0001	0.97396 0.0001	0.97468 0.0001	0.97930 0.0001	-0.87570 0.0009	0.97498 0.0001	0.97367 0.0001	0.96557 0.0001	0.99651 0.0001	0.92965 0.0001

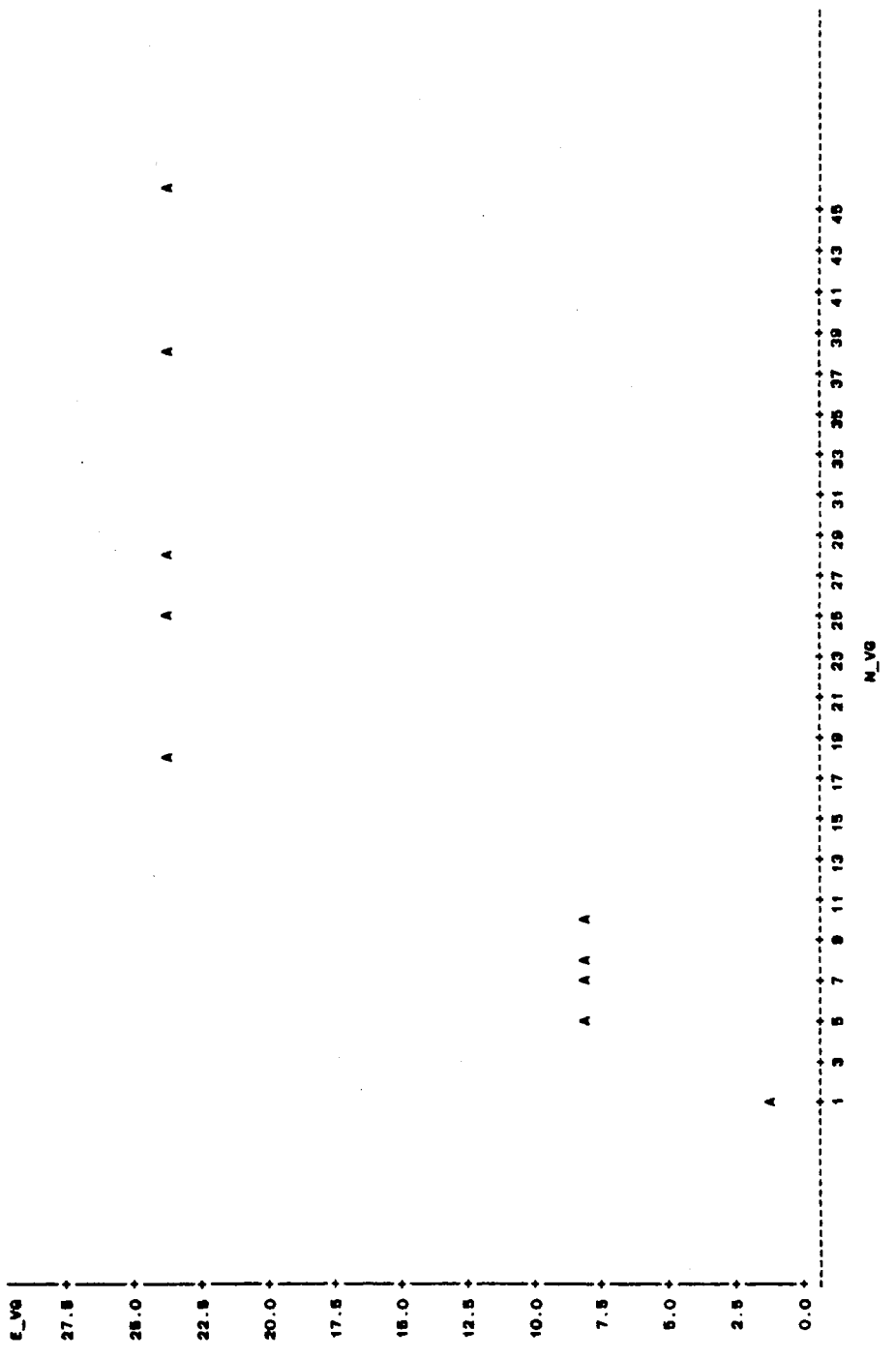
	N_BN1	N_BN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
E_N1	0.99636 0.0001	0.99185 0.0001	0.99180 0.0001	0.94587 0.0001	0.81310 0.0042	-0.27703 0.4384	0.83390 0.0027	0.99694 0.0001	0.95945 0.0001
E_N2	0.99497 0.0001	0.99766 0.0001	0.99618 0.0001	0.95286 0.0001	0.81850 0.0038	-0.25830 0.4712	0.84987 0.0018	0.99998 0.0001	0.97301 0.0001
E_BN1	0.99649 0.0001	0.99428 0.0001	0.99377 0.0001	0.94865 0.0001	0.81529 0.0040	-0.27120 0.4485	0.83948 0.0024	0.99848 0.0001	0.96433 0.0001
E_BN2	0.99216 0.0001	0.99861 0.0001	0.99639 0.0001	0.95452 0.0001	0.81962 0.0037	-0.24800 0.4897	0.85642 0.0016	0.99949 0.0001	0.97805 0.0001
E_V	0.99459 0.0001	0.99792 0.0001	0.99632 0.0001	0.95324 0.0001	0.81878 0.0038	-0.25657 0.4743	0.85107 0.0018	1.00000 0.0001	0.97396 0.0001
E_D	0.99483 0.0001	0.99787 0.0001	0.99671 0.0001	0.95460 0.0001	0.82125 0.0036	-0.25792 0.4719	0.84889 0.0019	0.99998 0.0001	0.97468 0.0001
E_E	0.99103 0.0001	0.99862 0.0001	0.99618 0.0001	0.95475 0.0001	0.81872 0.0037	-0.24481 0.4954	0.85817 0.0015	0.99906 0.0001	0.97930 0.0001
E_L	-0.96181 0.0001	-0.93306 0.0001	-0.93784 0.0001	-0.88491 0.0007	-0.76273 0.0103	0.32950 0.3525	-0.74693 0.0130	-0.84730 0.0001	-0.87570 0.0009
E_VG	0.83549 0.0001	0.96709 0.0001	0.95993 0.0001	0.92945 0.0001	0.79599 0.0059	-0.16868 0.6408	0.86822 0.0011	0.95814 0.0001	0.97498 0.0001
E_N	0.99463 0.0001	0.99790 0.0001	0.99631 0.0001	0.95321 0.0001	0.81875 0.0038	-0.25674 0.4740	0.85095 0.0018	1.00000 0.0000	0.97367 0.0001
E_NE	0.99643 0.0001	0.99484 0.0001	0.99421 0.0001	0.94932 0.0001	0.81581 0.0040	-0.26959 0.4513	0.84092 0.0023	0.99881 0.0001	0.96557 0.0001
N_N1	0.96585 0.0001	0.99066 0.0001	0.98418 0.0001	0.97245 0.0001	0.85766 0.0015	-0.18947 0.6001	0.82567 0.0033	0.97825 0.0001	0.99651 0.0001

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 10

	N_SN1	N_SN2	N_V	N_D	N_E	N_L	N_VG	N_N	N_NE
N_N2	0.88182 0.0007	0.91609 0.0002	0.88441 0.0007	0.78088 0.0064	0.59055 0.0723	0.15194 0.6752	0.86766 0.0011	0.90586 0.0003	0.92965 0.0001
N_SN1	1.00000 0.0000	0.99593 0.0001	0.99072 0.0001	0.94802 0.0001	0.81821 0.0036	-0.27682 0.4388	0.92072 0.0036	0.99463 0.0001	0.99866 0.0001
N_SN2	0.99593 0.0001	1.00000 0.0000	0.99436 0.0001	0.96124 0.0001	0.81462 0.0041	-0.24278 0.4991	0.96522 0.0012	0.99790 0.0001	0.97807 0.0001
N_V	0.99072 0.0001	0.99436 0.0001	1.00000 0.0000	0.97288 0.0001	0.86310 0.0013	-0.28163 0.4136	0.92277 0.0035	0.99631 0.0001	0.97817 0.0001
N_D	0.94802 0.0001	0.96124 0.0001	0.97288 0.0001	1.00000 0.0000	0.94436 0.0001	-0.38209 0.2758	0.75188 0.0122	0.95321 0.0001	0.95704 0.0001
N_E	0.81821 0.0036	0.81462 0.0041	0.86310 0.0013	0.94436 0.0001	1.00000 0.0000	-0.49176 0.1486	0.54927 0.1001	0.81875 0.0036	0.84086 0.0022
N_L	-0.27682 0.4388	-0.24278 0.4991	-0.28163 0.4136	-0.38209 0.2758	-0.49176 0.1486	1.00000 0.0000	-0.06811 0.8733	-0.25674 0.4740	-0.13439 0.7113
N_VG	0.92072 0.0036	0.96522 0.0012	0.92277 0.0035	0.75188 0.0122	0.54927 0.1001	-0.06811 0.8733	1.00000 0.0000	0.95095 0.0018	0.92318 0.0034
N_N	0.99463 0.0001	0.99790 0.0001	0.99631 0.0001	0.95321 0.0001	0.81875 0.0036	-0.25674 0.4740	0.95095 0.0018	1.00000 0.0000	0.97387 0.0001
N_NE	0.99866 0.0001	0.97807 0.0001	0.97817 0.0001	0.95704 0.0001	0.84086 0.0022	-0.13439 0.7113	0.92318 0.0034	0.97387 0.0001	1.00000 0.0000

PLOT OF E_N*N_N LEGEND: A = 1 OBS, B = 2 OBS, ETC.
PLOT OF E_N*N_NE SYMBOL USED IS *




SAS
12:32 WEDNESDAY, JUNE 7, 1989
PLOT OF E_VG*N_VG LEGEND: A = 1 OBS, B = 2 OBS, ETC.
E_VG
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45
N_VG

VITA

Budy Tjahjo

Candidate for the Degree of

Master of Science

Thesis: A STUDY OF THE EFFECT OF THE OBJECT-ORIENTED PROGRAMMING PARADIGM ON PROGRAMMING DESIGN COMPLEXITY USING ADA, MODULA-2, AND C++

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Surabaya, Indonesia, April 5, 1962, the son of Hendro Tjahjo and Ida Sutjiati Tjipto.

Education: Graduated from Sancta Maria High School, Surabaya, Indonesia, in August 1981; received Bachelor of Science Degree in Computing and Information sciences at Oklahoma State University in July, 1985; completed requirements for the Master of Science degree at Oklahoma State University in December 1989.

Professional Experience: Programmer, Department of Agricultural, Oklahoma State University, August, 1988 to June, 1989.