



NRL/MR/6110--04-8774

Isotope Ratio Mass Spectrometry Data Processing Software: Multivariate Statistical Methods for Hydrocarbon Source Identification and Comparison

THOMAS J. BOYD
RICHARD B. COFFIN

*Chemical Dynamics and Diagnostics Branch
Chemistry Division*

April 29, 2004

Approved for public release; distribution is unlimited.

20040513 053

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 29 April 2004		2. REPORT TYPE Final report		3. DATES COVERED (From - To) July 2003-December 2003	
4. TITLE AND SUBTITLE Isotope Ratio Mass Spectrometry Data Processing Software: Multivariate Statistical Methods for Hydrocarbon Source Identification and Comparison				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Thomas J. Boyd and Richard B. Coffin				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER 61-7800-G3	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory, Code 6114 4555 Overlook Avenue, SW Washington, DC 20375-5320				8. PERFORMING ORGANIZATION REPORT NUMBER NRL/MR/6110--04-8774	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Sea Systems Command 1333 Isaac Hull Avenue, S.E. Washington Navy Yard, DC 20376				10. SPONSOR / MONITOR'S ACRONYM(S)	
				11. SPONSOR / MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The IRMS Data Processing software package is designed to allow easy stable isotope data entry and multivariate data analysis. When comparing two or more hydrocarbon samples using compound-specific isotope ratio mass spectrometry, an analyst obtains multiple data variables for each sample. Multivariate statistics allows rigorous comparison(s) to determine if the samples are in fact different and if so, how closely related they are. This software uses three main types of data analyses: Multiple Analysis of Variance (MANOVA), Principal Components Analysis (PCA), and Cluster Analysis. The layout is a standard Windows interface which should be usable to anyone familiar with modern operating system software.					
15. SUBJECT TERMS Software; Stable isotope ratios; Statistical analysis; Multiple Analysis of Variance; Principal Components Analysis; Hierarchical clustering; Data table					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UL	18. NUMBER OF PAGES 145	19a. NAME OF RESPONSIBLE PERSON Thomas J. Boyd
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (include area code) (202) 404-6424

CONTENTS

Introduction.....	1
Goals	2
Methods.....	2
Software	3
Introduction.....	3
Program Introduction	4
Data Table Setup.....	4
Statistical Analyses	12
Manova	13
PCA.....	15
Cluster.....	17
Export, Copying, Printing, and Saving Data and Graphics	19
Exiting the Application.....	20
Literature Cited.....	21
Appendix I. Code Listing.....	23

ISOTOPE RATIO MASS SPECTROMETRY DATA PROCESSING SOFTWARE: MULTIVARIATE STATISTICAL METHODS FOR HYDROCARBON SOURCE IDENTIFICATION AND COMPARISON

INTRODUCTION

Oil spills present a significant problem for domestic Naval operations. Annual cleanup costs approach \$10M with nearly 1,600 spills totaling 255,000 gallons reported from FY97 through FY03. With these spills, the Navy is in violation of the Clean Water Act, which prohibits discharge of oil in amounts sufficient to produce a visible sheen on the water surface. Although the Navy is exempt from fines and penalties from oil spills, environmental ramifications have attracted high-level Congressional, State, and local concern. States with large Naval fleet presence such as California, Washington, Virginia and Texas have shown particular interest in Naval fuel spills. The lack of measurable progress in reducing the number and volume of spills may impact the Navy's Public Vessel Exemption, resulting in fines, penalties and remediation costs that would total in the millions annually.

In March 1999, the Naval Sea Systems Command, as directed by the CNO, prepared a Shipboard Oil Spill Prevention Initiative plan. The plan was based on actual NAVSEA Shipboard analysis and the results of a workshop held in Norfolk, VA in August 1999. The initial plan was aimed at reducing or eliminating fuel spills by applying lessons from known causes. To this end, the FUEL ID initiative was created to provide identification of spill versus non-spill oil signatures in the environment. Compound-specific carbon isotope analysis (CSIA) coupled to multivariate statistics was identified as a robust means of determining similarity between unknown spill oils and those from Naval sources.

Frequently multiple sources exist and complex mixing and transport result in uncertain assessment and organization of remedial action. A number of fingerprinting approaches have been developed to determine source and fate of hydrocarbons, the most common of which is to determine the relative concentrations of individual hydrocarbons in a mixture. The major drawback of this approach is that it does not take into account weathering activities (i.e. biological, physical) which might preferentially remove certain components of the mixture. Stable isotope analyses of elements provides the ability to identify the sources and fate in complex mixtures of environmental organic matter by targeting a concentration-independent chemical property of each contaminant in a mixture. Isotope analysis of carbon, nitrogen and sulfur pools has provided a more thorough understanding of organic matter sources and cycling in a variety of ecosystems (Peterson and Fry, 1987; Fry, 1986; Coffin and Cifuentes, 1999). Further development of isotope ratio methodology has provided the ability to identify cycling of carbon at a molecular level (Coffin et al. 1990; Silfer et al. 1991; Meier-Augenstein, 1995; Hullar et al., 1996) allowing identification of specific microbial roles in the biogeochemical cycling of carbon and nitrogen. In addition, this approach has provided the capacity to use stable carbon isotope analysis ($\delta^{13}\text{C}$) to assist in development and interpretation of bioremediation strategies for ecosystems that are contaminated with organic chemicals (Aggarwal and Hinchee, 1991; Trust et al., 1995; Coffin et al., 1997).

The recent coupling of gas chromatography (GC) to transfer individual compounds, combusted inline, to the isotope ratio mass spectrometer (IRMS) provides a two dimensional ability to identify individual contaminant sources (e.g. Hammer et al. 1998).

Preliminary experiments demonstrate that the carbon isotope signature in 2-, 3-, 4-, 5-ring PAHs is stable to vaporization, photolytic decomposition and microbial degradation (O'Malley et al., 1994). If contaminant sources have a broad range in $\delta^{13}\text{C}$ it is possible to determine the contribution of a source to the total loading. With $\delta^{13}\text{C}$ analysis the percent of vehicular emissions and crank case oil in the total PAH loading was estimated in the St. John's Harbour, Newfoundland (O'Malley et al., 1996). In a similar study using $\delta^{13}\text{C}$ for analysis of benzene, toluene, ethylbenzene and xylene (BTEX) multiple petroleum sources were shown to be present in groundwater that was thought to be contaminated with one source (Kelley et al. 1997). Other recent research provides further support for the application of carbon isotope analysis to trace the contaminant sources. This approach has been applied in the tracking of nitroaromatic compounds (Coffin et al. 2001), PCE and TCE (Lollar et al. 2001), and jet fuels (Landmeyer et al. 1996). This research has initiated the application of carbon isotope analysis to assess organic contaminant sources in ecosystems.

GOALS

1. Develop the software to survey carbon isotope ratio data for determination of contaminant sources.
2. Initiate a stable carbon isotope facility at the Norfolk Navy Base to determine the source(s) of petroleum spills.

METHODS

This project applies the recent development in stable carbon isotope analysis to trace fuel sources at the Norfolk Navy Base. The preliminary step in this project was to use existing and contemporaneously-gathered data to develop a hydrocarbon stable carbon analysis software application. This application consists of a data entry module, data analysis module and a reporting module. The data entry module allows users to import excel data, or enter stable carbon isotope data directly into the application. A user will then be able to perform a series of statistical analysis (as described below) to determine the similarities between hydrocarbon samples. The reporting module displays and can "export" the results of the analysis for inclusion in standard documenting formats (i.e. Word®, Powerpoint®, etc). The analysis module processes data in a number of ways. One of the difficulties in interpreting data from isotope analyses is that there are more than two variables, negating a simple, direct analysis of variance. In the data entry module a series of alkanes and their $\delta^{13}\text{C}$ values will be entered. In this way, there will at least eight separate variables (i.e. C_{10} , C_{11} , C_{12} , etc) per sample. Data with multiple observations and multiple variables lends itself to multivariate analysis. In developing the analysis software module, the Matlab® multivariate statistical toolbox was used.

The first analysis is a MANOVA or multiple analysis of variance. This analysis allows one to determine if there are statistically significant differences between two samples with multiple variables. Data output from this test provides a probability that the two samples are the same. Generally, if the *P* (or probability) value is less than 0.05, there is

only a 5% chance that the two samples are the same. The analysis module will allow the user to select the desired probability reporting (i.e. 5% or 1%) as a screening tool. The actual probabilities are calculated and transferred to the reporting module. Aside from determining if two sources are "different," it will also be of use to determine how similar two sources are. For instance, if two sources intermingle, the resulting mixture might be "different" from each of its parent sources; however it might be closely related to both. Principal components or factor analysis (PCA) can help an investigator determine how closely related two samples are by simplifying the factors controlling variability. By plotting the first two factors against one another, samples can be visualized based on their relatedness. Each factor is given a weighting as to how important it is in describing the variability found in the original data. Cluster analysis is another multivariate means to determine the relatedness of samples. This analysis does not try to "simplify" the variability between samples, and therefore must be interpreted in light of the cophenetic correlation coefficient (which in the case of the test data set used here was too low for acceptable results). Matlab[®] has a number of protocols to fine tune data for inclusion in this analysis. The reporting module collates information from the analysis module and outputs data in a report format. The output is exportable to standard formats (i.e. Word[®], Powerpoint[®], PDF[®], etc) as well as printable on any Windows-installed printer.

The underlying statistics for the application are derived from the Matlab[®] computing language using the Matlab[®] compiler which allows Matlab[®] code to be converted to C/C++. Although the Matlab[®] environment includes a graphical user interface development module, statistical routines were exported to C++ code and compiled into dynamic libraries that were included in a program developed within the Microsoft[®] Visual Studio.net environment. In this manner, the "standard" Windows[®] interface is used for the finished product. The separate modules (data entry, data analysis, reporting) work together within an overall stand-alone Windows[®] application.

THE SOFTWARE

I. Introduction

The IRMS Data Processing software package (IRMS-DP) is designed to allow easy stable isotope data entry and multivariate data analysis. When comparing two or more hydrocarbon samples using compound-specific isotope ratio mass spectrometry, an analyst obtains multiple data variables for each sample. For instance with volatile samples, one may be able to separate benzene, toluene, ethyl-benzene, *p*-xylene, *o*-xylene and *m*-xylene and obtain a stable isotope ratio for each. Multivariate statistics allows rigorous comparison(s) to determine if the samples are in fact different and if so, how closely related they are.

This software uses three main types of data analyses: Multiple Analysis of Variance (MANOVA), Principal Components Analysis (PCA), and Cluster Analysis. In data sets with multiple variables, it is desirable to determine if the means of two samples are significantly different. A multiple analysis of variance (MANOVA) can be used to produce probability values. A P value of 0.01 essentially means that one can be 99%

certain that chance alone would not lead to the differences seen between sample means.

In data sets with multiple variables, groups of variables often behave similarly. More than one variable may in fact be describing the same principle of the system. PCA attempts to simplify a multivariate data set by replacing a group of variables with a single new variable, called a principal component. Each principal component is a linear combination of the original variables. The variance of each principal component is the maximum among all possible choices. The analysis provides information as to how much of the original variance is represented by each principal component. Therefore, when the primary components are graphed against one-another, data sets that are highly similar will plot together, while dissimilar data sets will occupy different spaces on a graph. The result of placing the scores in a new coordinate system allows visualizing the data.

In addition to PCA analysis, clustering analysis can be used to determine a relative 'distance' between relations in multivariate data. This would be analogous to plotting a family tree and using one inch to represent each generation of distance between progenitors and progeny. The length of vertical lines in clusters is indicative of the 'distance' of relatedness between samples.

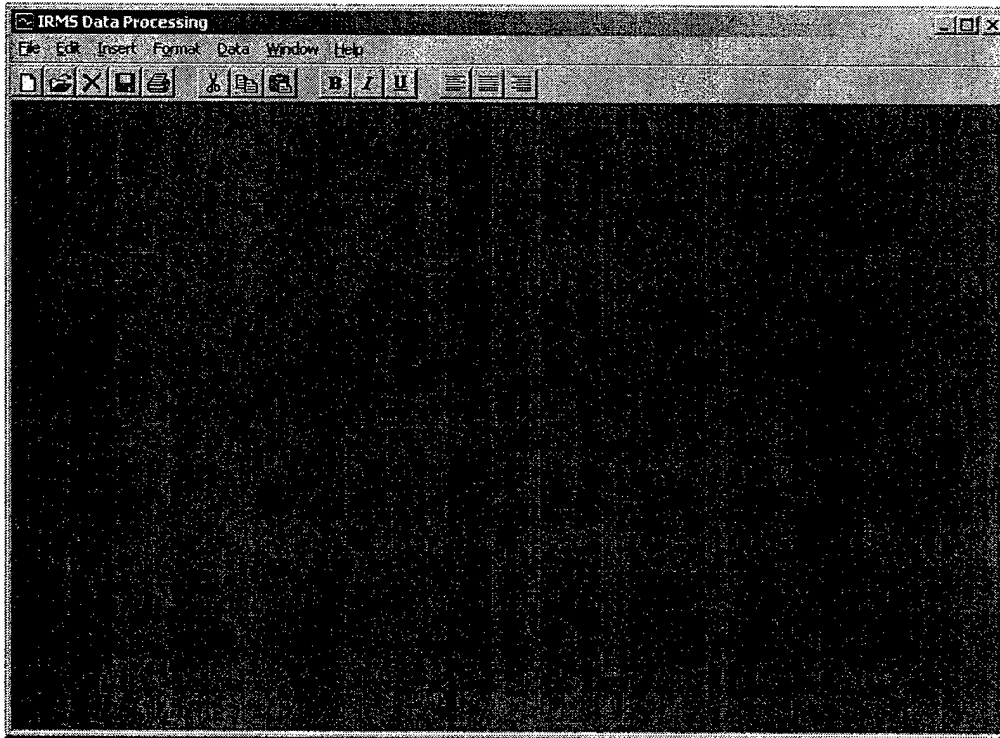
II. Program Introduction

IRMS-DP is meant to be similar to any windows spreadsheet software for data entry. Each step in data entry and subsequent analysis is menu driven allowing a "non-statistician" to use the software effectively. The user is asked how many replicates will be entered (i.e. how many replicate sample runs) and whether he/she wishes to name each variable in the data grid. Naming or not naming variables will not impact the data analysis so this feature is provided solely for the convenience of the user. Once the data grid is created, the user enters the sample name (or SampleID) for each sample and the individual stable isotope ratios for each compound (variable). These can be manually entered or pasted into the grid from a text or spreadsheet application. **One data grid should be made for each set of measurements with the same number of variables.** For example, if seven hydrocarbons (variables), such as nonane, decane, undecane, dodecane, tridecane, tetradecane, pentadecane, and hexadecane were determined for 8 samples, but in 4 samples, tetradecane was not resolvable, the 4 samples (without tetradecane) must be placed in a separate data table for analysis. **Only samples with the same number of variables can be directly compared to one another with this program's statistical techniques.**

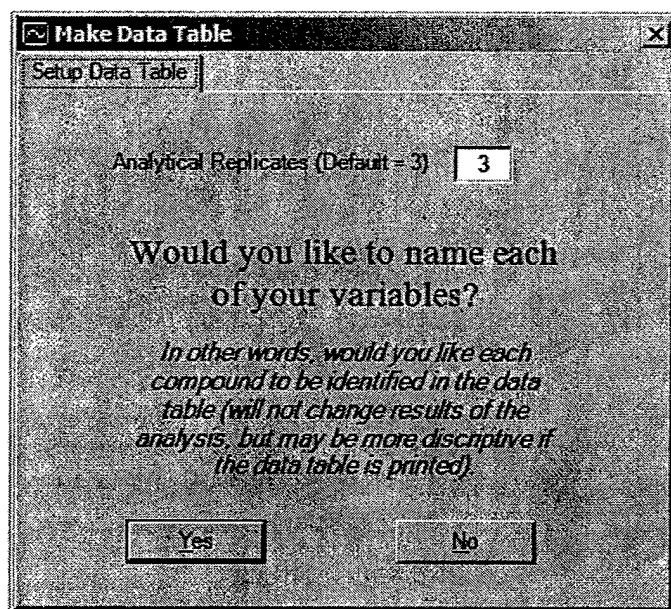
Once the data grid is complete, the user can choose any of the three main statistical tests and receive results. The results are displayed in graphs and in associated text box(s) so the user can "keep" the most useable data and results. Data, results and graphs are exportable and savable to be portable between IRMS-PD and presentation/graphics software.

III. Data Table Setup

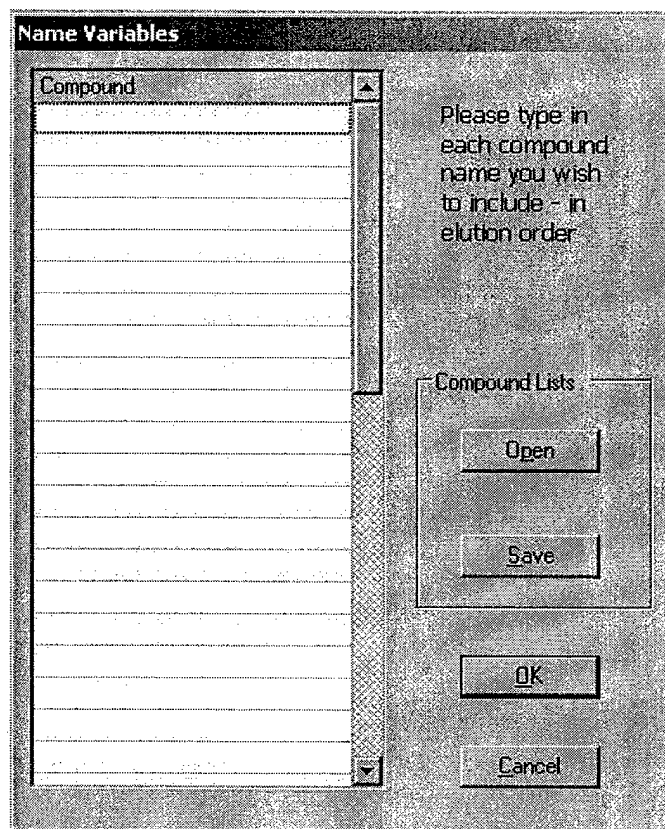
The Data Table or Grid is the first entry step for using IRMS-DP. Upon opening the program, the user is presented with a "blank" program workspace:



The "standard" windows menus are available as well as a toolbar representing shortcuts to commonly used menu items. The first step in performing an analysis is to create or open a Data Table. Under the file menu, there is a choice for New or Open. These choices are also represented by the first two toolbar buttons. If the user chooses to create a new Data Table, he/she is presented with the following screen:



The Make Data Table dialog box requires the user to select the number of replicates to be entered for each sample. The default is three (3). This can be changed to any number the user wishes. However, data must be entered for all of the replicates specified. Therefore, the user should select the lowest number of replicates found in the group of samples to be analyzed. The user is also asked if he/she wishes to name the variables. This is not necessary, but makes manual data entry easier (with named column headings). If the user chooses to name variables, he/she will be provided with the following dialog box:



The image shows a software dialog box titled "Name Variables". On the left is a large grid with a header row labeled "Compound". The grid is currently empty. To the right of the grid, there is a text instruction: "Please type in each compound name you wish to include - in elution order". Below this instruction is a section titled "Compound Lists" which contains two buttons: "Open" and "Save". At the bottom of the dialog box are two more buttons: "OK" and "Cancel".

Variables (i.e. individual compounds) can be entered just as one would enter data into a spreadsheet. Once entered, the variable names can be saved to a text file for use in subsequent analyses. Alternatively, if the same compounds were used in a previous analysis and saved, the file can be opened to populate the Compound grid:

Name Variables

Compound
nonane
decane
undecane
dodecane
tridecane
tetradecane
pentadecane
hexadecane
heptadecane
octadecane
nonadecane

Please type in each compound name you wish to include - in elution order

Compound Lists

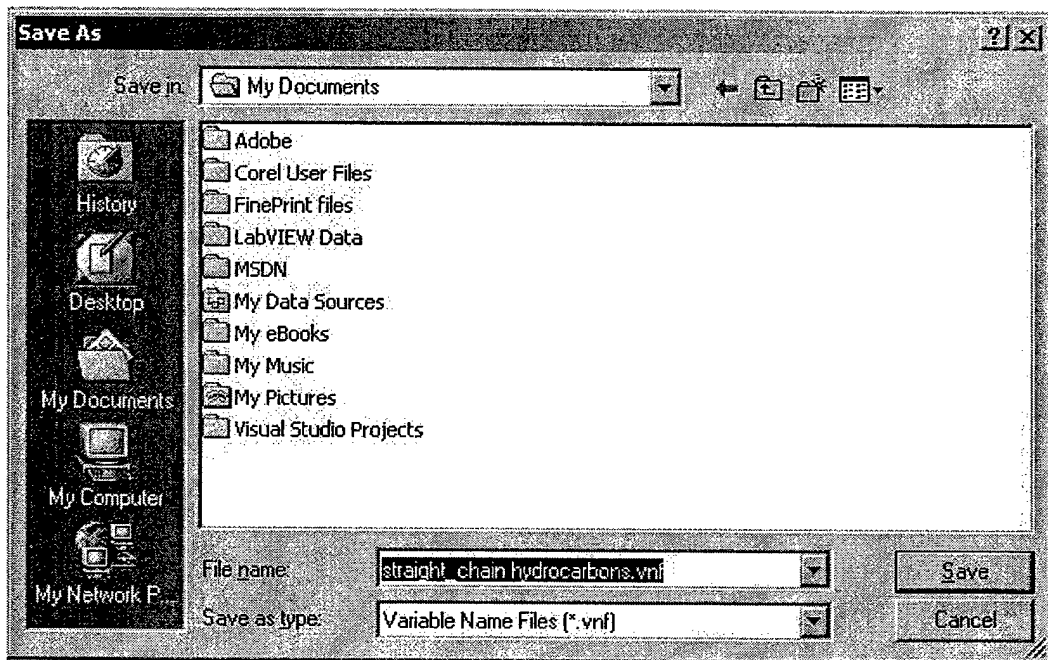
Open

Save

OK

Cancel

Above is listing of straight-chain hydrocarbons that can be analyzed by compound-specific isotope ratio mass spectrometry. If Save is clicked, the file can be stored for future use:



When OK is pressed (or No in the Make Table dialog box), a Data Table is created for the user:

The screenshot shows the 'IRMS Data Processing' application window. It features a menu bar (File, Edit, Insert, Format, Data, Window, Help) and a toolbar with various icons. The main area displays 'Data Table 1' with the following structure:

	SampleID	Replicate	nonane	decane	undecane	dodecane	tridecane	tetradecane	pentadecane
1		1							
2		2							
3		3							
4		1							
5		2							
6		3							
7		1							
8		2							
9		3							
10		1							
11		2							
12		3							
13		1							
14		2							
15		3							
16		1							
17		2							
18		3							
19		1							
20		2							
21		3							
22		1							
23		2							
24		3							
25		1							
26		2							

After the Data Table is created, the user can enter the SampleID and individual measurements for each compound and replicate in the group of samples:

IRMS Data Processing

File Edit Insert Format Data Window Help

Data Table 1

SampleID	Replicate	nonane	decane	undecane	dodecane	hexadecane
1 Sample 1	1	-26.05	-25.81	-24.06	-25.05	-25.44
2	2	-26.18	-25.94	-24.05	-24.61	-25.69
3	3	-26.15	-26.51	-23.97	-24.45	-23.57
4 Sample 2	1	-25.24	-24.84	-23.57	-24.59	-25.08
5	2	-25.55	-25.11	-25.04	-24.32	-24.67
6	3	-25.35	-24.8	-23.18	-23.9	-24.02
7 Sample 3	1	-26.4	-25.86	-25.27	-24.05	-23.14
8	2	-26.05	-25.3	-23.46	-21.74	-24.59
9	3	-25.55	-25.11	-25.04	-24.32	-24.67
10 Sample 4	1	-25.24	-24.84	-23.57	-24.59	-25.08
11	2	-25.55	-25.11	-25.04	-24.32	-24.67
12	3	-26.05	-25.81	-24.06	-25.05	-25.44
13 Sample 5	1	-28.23	-25.83	-24.1	-26.23	-23.24
14	2	-28.33	-25.88	-24.23	-26.34	-23.03
15	3	-27.99	-25.75	-24.01	-26.5	-23.3
16 Sample 6	1	-24.12	-26.23	-27.77	-28.21	-23.33
17	2	-24.33	-26.29	-27.69	-28.01	-24.01
18	3	-23.99	-26.15	-27.8	-28.3	-23.56
19	1					
20	2					
21	3					
22	1					
23	2					

The program checks each isotope value to make sure it is >-100 and < 100 as a check for the user. Cells in the Replicate column are locked because the statistical methods rely on replicate analyses (of known and fixed value) for processing. User preferences for text style and cell colors can be made using the Format > Cells menu and toolbar icons:

The screenshot shows the 'IRMS Data Processing' window. It contains a menu bar (File, Edit, Insert, Format, Data, Window, Help) and a toolbar with various icons. Below the toolbar is a 'Data Table 1' window containing a table with the following data:

SampleID	Replicate	nonane	decane	undecane	dodecane	tridecane
1	Sample 1	1	-26.05	-25.81	-24.06	-25.05
2		2	-26.18	-25.94	-24.05	-24.61
3		3	-26.15	-26.51	-23.97	-24.45
4	Sample 2	1	-25.24	-24.84	-23.57	-24.59
5		2	-25.55	-25.11	-25.04	-24.32
6		3	-25.35	-24.8	-23.18	-23.9
7	Sample 3	1	-26.4	-25.86	-25.27	-24.05
8		2	-26.05	-25.3	-23.46	-21.74
9		3	-25.55	-25.11	-25.04	-24.32
10	Sample 4	1	-25.24	-24.84	-23.57	-24.59
11		2	-25.55	-25.11	-25.04	-24.32
12		3	-26.05	-25.81	-24.06	-25.05
13	Sample 5	1	-28.29	-25.83	-24.1	-26.23
14		2	-28.33	-25.88	-24.23	-26.34
15		3	-27.99	-25.75	-24.01	-23.3
16	Sample 6	1	-24.12	-26.23	-27.77	-28.21
17		2	-24.33	-26.29	-27.69	-28.01
18		3	-23.99	-26.15	-27.8	-28.3
19		1				
20		2				
21		3				
22		1				
23		2				

IV. Statistical Analyses

Statistical analyses are located under the Data menu. The user can choose between Manova, PCA and Cluster. Regardless of the choice, the user is presented with a dialog to select samples to be included in the analysis:

IRMS Data Processing								
File Edit Insert Format Data Window Help								
Process Statistics Manova PCA Cluster								
Data Table 1								
	SampleID	Replicate	nonane	decane	undecane	dodecane	hexadecane	
1	Sample 1	1	-26.05	-25.81	-24.06	-25.05	-25.44	
2		2	-26.18	-25.94	-24.05	-24.61	-25.69	
3		3	-26.15	-26.51	-23.97	-24.45	-23.57	
4	Sample 2	1	-25.24	-24.84	-23.57	-24.59	-25.08	
5		2	-25.55	-25.11	-25.04	-24.32	-24.67	
6		3	-25.35	-24.8	-23.18	-23.9	-24.02	
7	Sample 3	1	-26.4	-25.86	-25.27	-24.05	-23.14	
8		2	-26.05	-25.3	-23.46	-21.74	-24.59	
9		3	-25.55	-25.11	-25.04	-24.32	-24.67	
10	Sample 4	1	-25.24	-24.84	-23.57	-24.59	-25.08	
11		2	-25.55	-25.11	-25.04	-24.32	-24.67	
12		3	-26.05	-25.81	-24.06	-25.05	-25.44	
13	Sample 5	1	-28.23	-25.83	-24.1	-26.23	-23.24	
14		2	-28.33	-25.88	-24.23	-26.34	-23.03	
15		3	-27.99	-25.75	-24.01	-26.5	-23.3	
16	Sample 6	1	-24.12	-26.23	-27.77	-28.21	-23.33	
17		2	-24.33	-26.29	-27.69	-28.01	-24.01	
18		3	-23.99	-26.15	-27.8	-28.3	-23.56	
19		1						
20		2						
21		3						
22		1						
23		2						

IRMS Data Processing

File Edit Insert Format Data Window Help

Data Table 1 Manova Output 1/6/2004 8:29:37 AM

Multiple analysis of variance completed successfully.

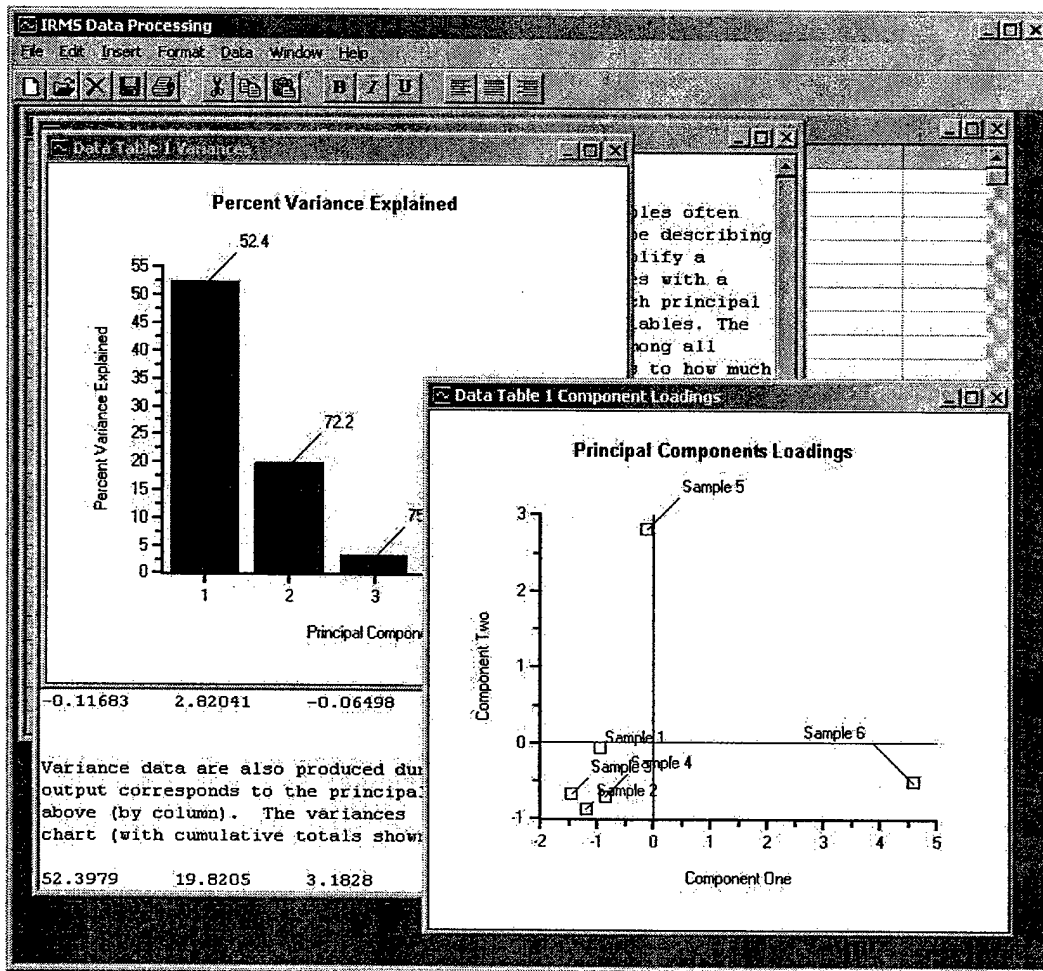
In data sets with multiple variables, it is desirable to determine if the means of two samples are significantly different. A multiple analysis of variance (MANOVA) can be used to produce probability values. A P value of 0.01 essentially means that one can be 99% certain that chance alone would not lead to the differences seen between sample means. In this analysis, one must have a 'square' matrix. Therefore, the original data is expanded using a random number generator to produce the proper matrix dimensions.

The following table shows each test (Sample 1 vs Sample 2) and the P value. A P value of less than 0.05 indicates a significant difference.

Sample 1	Sample 2	P value
Sample 1	Sample 2	< 0.001
Sample 1	Sample 3	< 0.001
Sample 1	Sample 4	< 0.001
Sample 1	Sample 5	< 0.001
Sample 1	Sample 6	< 0.001
Sample 2	Sample 3	< 0.001
Sample 2	Sample 4	< 0.001
Sample 2	Sample 5	< 0.001
Sample 2	Sample 6	< 0.001
Sample 3	Sample 4	< 0.001
Sample 3	Sample 5	< 0.001
Sample 3	Sample 6	< 0.001
Sample 4	Sample 5	< 0.001
Sample 4	Sample 6	< 0.001
Sample 5	Sample 6	< 0.001

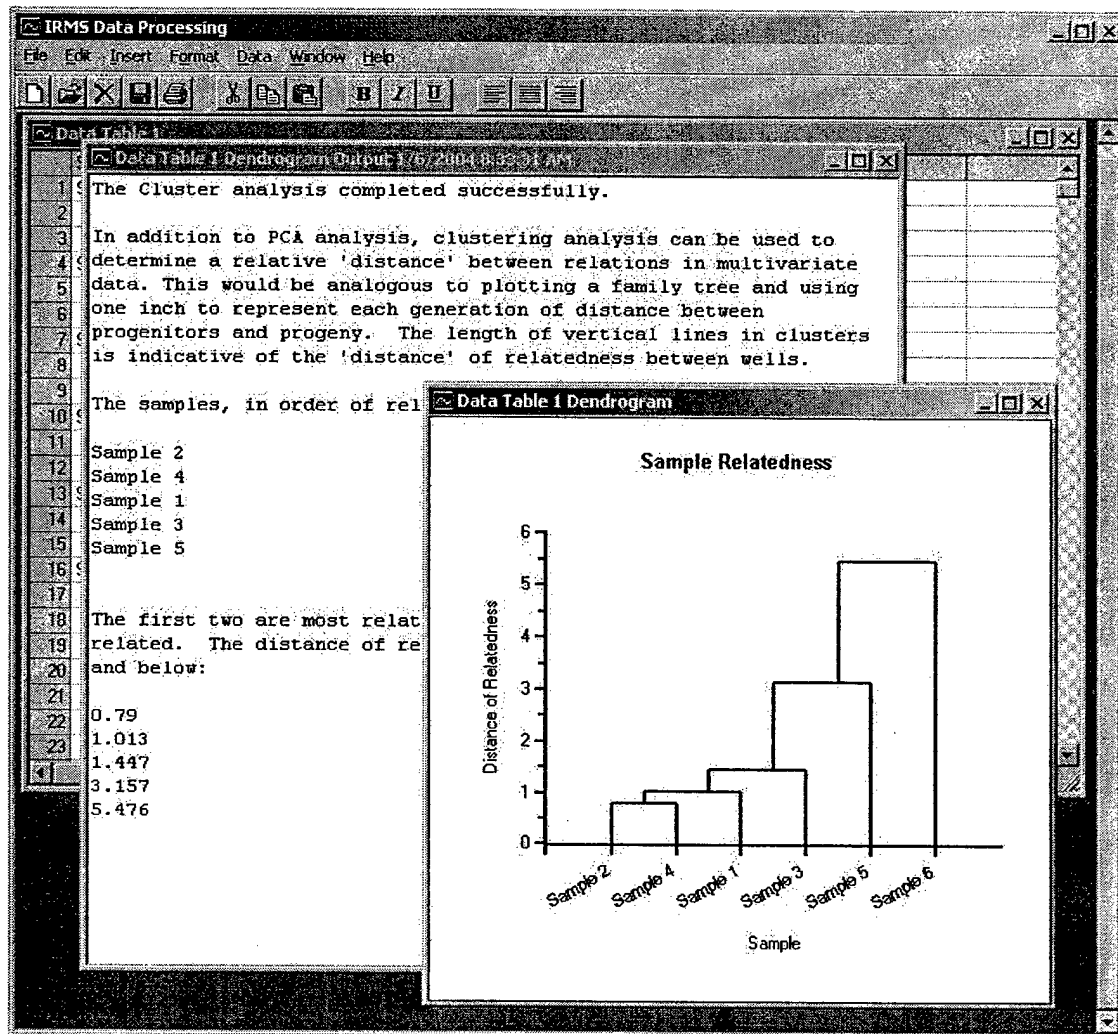
B. PCA. PCA is a method by which variability in data is represented by a "new" series of variables. These new components represent a principle of the variability in the original data set. The variability for each principal component is represented in a generated table. Typically, the first two components explain 70% or more of the intersample variability. For this reason, these components are graphed against one another so that data can be clustered into "like" samples. Samples that line up in Component One (i.e. have similar X distribution) are likely quite similar (if the first Component accounts for >50% of the variability). Samples that line up in the Component Two (i.e. have similar Y distribution) are also likely to be similar (because the second Component accounts for the second most variability). Those samples that cluster together when X is plotted again Y should therefore be very closely related. When PCA is selected in IRMS-DP, Component One is graphed against Component Two and the Variability attributable to each Component is also graphed. In addition, a text output is provided with an explanation of the graphs:

IRMS Data Processing							
File Edit Insert Format Data Window Help							
Process Statistics PCA Cluster							
Data Table 1							
SampleID	Replicate	nonane	decane	undecane	dodecane	hexadecane	
1 Sample 1	1	-26.05	-25.81	-24.06	-25.05	-25.44	
2	2	-26.18	-25.94	-24.05	-24.61	-25.69	
3	3	-26.15	-26.51	-23.97	-24.45	-23.57	
4 Sample 2	1	-25.24	-24.84	-23.57	-24.59	-25.08	
5	2	-25.55	-25.11	-25.04	-24.32	-24.67	
6	3	-25.35	-24.8	-23.18	-23.9	-24.02	
7 Sample 3	1	-26.4	-25.86	-25.27	-24.05	-23.14	
8	2	-26.05	-25.3	-23.46	-21.74	-24.59	
9	3	-25.55	-25.11	-25.04	-24.32	-24.67	
10 Sample 4	1	-25.24	-24.84	-23.57	-24.59	-25.08	
11	2	-25.55	-25.11	-25.04	-24.32	-24.67	
12	3	-26.05	-25.81	-24.06	-25.05	-25.44	
13 Sample 5	1	-28.23	-25.83	-24.1	-26.23	-23.24	
14	2	-28.33	-25.88	-24.23	-26.34	-23.03	
15	3	-27.99	-25.75	-24.01	-26.5	-23.3	
16 Sample 6	1	-24.12	-26.23	-27.77	-28.21	-23.33	
17	2	-24.33	-26.29	-27.69	-28.01	-24.01	
18	3	-23.99	-26.15	-27.8	-28.3	-23.56	
19	1						
20	2						
21	3						
22	1						
23	2						



C. Cluster. Clustering analysis can be used to determine a relative 'distance' between relations in multivariate data. The length of vertical lines in clusters is indicative of the 'distance' of relatedness between wells. When the Cluster analysis is selected, the data are analyzed and a dendrogram is presented to the user along with a text description of the graphic.

IRMS Data Processing								
File Edit Insert Format Data Window Help								
Process Statistics Manova PCA								
Data Table 1 Cluster								
SampleID	Replicate	nonane	decane	undecane	dodecane	hexadecane		
1 Sample 1	1	-26.05	-25.81	-24.06	-25.05	-25.44		
2	2	-26.18	-25.94	-24.05	-24.61	-25.69		
3	3	-26.15	-26.51	-23.97	-24.45	-23.57		
4 Sample 2	1	-25.24	-24.84	-23.57	-24.59	-25.08		
5	2	-25.55	-25.11	-25.04	-24.32	-24.67		
6	3	-25.35	-24.8	-23.18	-23.9	-24.02		
7 Sample 3	1	-26.4	-25.86	-25.27	-24.05	-23.14		
8	2	-26.05	-25.3	-23.46	-21.74	-24.59		
9	3	-25.55	-25.11	-25.04	-24.32	-24.67		
10 Sample 4	1	-25.24	-24.84	-23.57	-24.59	-25.08		
11	2	-25.55	-25.11	-25.04	-24.32	-24.67		
12	3	-26.05	-25.81	-24.06	-25.05	-25.44		
13 Sample 5	1	-28.23	-25.83	-24.1	-26.23	-23.24		
14	2	-28.33	-25.88	-24.23	-26.34	-23.03		
15	3	-27.99	-25.75	-24.01	-26.5	-23.3		
16 Sample 6	1	-24.12	-26.23	-27.77	-28.21	-23.33		
17	2	-24.33	-26.29	-27.69	-28.01	-24.01		
18	3	-23.99	-26.15	-27.8	-28.3	-23.56		
19	1							
20	2							
21	3							
22	1							
23	2							



V. Export, Copying, Printing, and Saving Data and Graphics

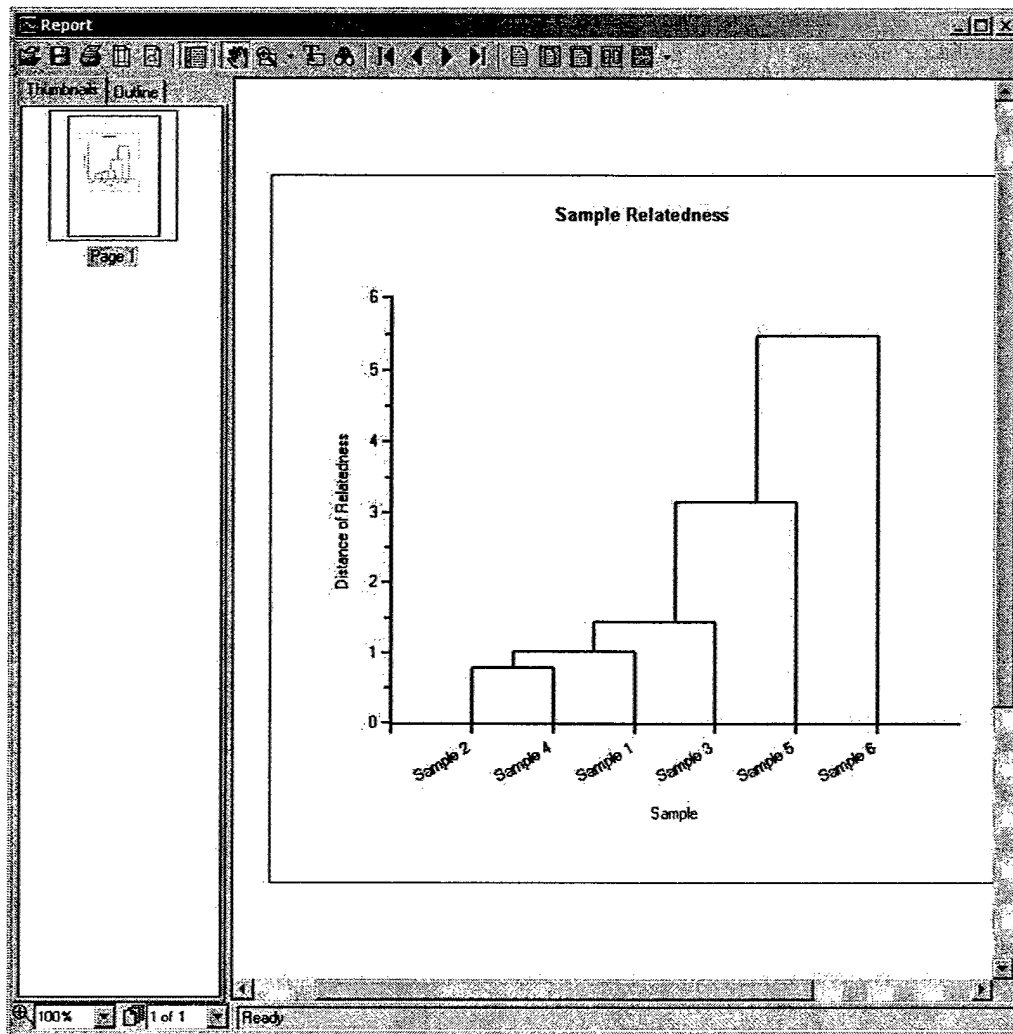
The IRMS-DP package can export Data Tables, graphs and text windows. The easiest way to export these objects is to use the Windows clipboard. Copy operations can be conducted through the Edit menu, the button bar, and the context menu (right-click menu) available for each open object.



Alternatively, Table, text and graphs can be exported (Save As) to comma separated values (CSV), plain text (TXT) or enhanced metafile (EMF) respectively. Save operations are available through the File menu or through context menu(s) on each

application sub-window.

Printing operations all use the same rendering engine. Each document type (Data Table, text, or graph) is converted to a print document and displayed to the user for additional print formatting. Pressing the Print icon in the preview toolbar sends the data to a selected printer:



VI. Exiting the Application

The application can be exited by pressing <Alt> F4, by clicking the X in the upper right-hand corner, or by selecting Exit under the File menu.

Literature Cited

- Aggarwal P. K., R. E. Hinchee. 1991. Monitoring in situ biodegradation of hydrocarbons by using stable carbon isotopes. *Environ Sci Technol* 25:1178-1180.
- Coffin, R. B., D. Velinsky, R. Devereux, Wm. Allen Price and L. Cifuentes. 1990. Stable carbon isotope analysis of nucleic acids to trace sources of dissolved substrate used by estuarine bacteria. *Appl. Environ. Microbiol.* 56:2012-2020.
- Coffin, R. B., L. A. Cifuentes, and P. M. Elderidge. 1994. The use of stable carbon isotopes to study microbial processes in estuaries. In: Lajtha, K. and R. Michener, eds., *Stable Isotopes in Ecology*, pp. 222-240.
- Coffin, R. B., L. A. Cifuentes and P. H. Pritchard. 1997. Effect of remedial nitrogen applications on algae and heterotrophic organisms on oil contaminated beaches in Prince William Sound, AK. *Mar. Environ. Res.* 1:27-39.
- Coffin, R. B. and L. A. Cifuentes. 1999. Examination of carbon and nitrogen sources responsible for anoxia in the Perdido Estuary, Florida using stable isotope ratios. *Estuaries*. 22:997-1006.
- Coffin, R. B., P. H. Miyares, C. A. Kelley, L. A. Cifuentes and C. M. Reynolds. (In Press) $\delta^{13}\text{C}$ and $\delta^{15}\text{N}$ Isotope Analysis of TNT: Two Dimensional Source Identification. *Environmental Toxicology and Chemistry*.
- Fry B. 1986. Sources of carbon and sulfur nutrition for consumers in three meromictic lakes of New York State. *Limnol Oceanogr* 31:79-88.
- Hammer B. T., C. A. Kelley, R. B. Coffin, L. A. Cifuentes, J. Mueller. 1998. $\delta^{13}\text{C}$ values of polycyclic aromatic hydrocarbon collected from two creosote-contaminated sites. *Chem Geol (Isotope Geoscience)*.158:43-58.
- Hullar M. A. J., B. Fry, B. J. Peterson, R. T. Wright. 1996. Microbial utilization of estuarine dissolved organic carbon: A stable isotope tracer approach tested by mass balance. *Appl Environ Microbiol* 62:2489-2493.
- Kelley, C. A., B. A. Trust and R. B. Coffin. 1997. Tracing BTEX sources and transport in contaminated groundwater environments with GC/IRMS/ITMS. *Environ. Sci. Technol.* 31:2469-2472.
- Landmeyer, J. E., D. A. Vroblesky and F. H. Chapelle. 1996. Stable Carbon Isotope Evidence of Biodegradation Zonation in a Shallow Jet-Fuel Contaminated Aquifer. *Environ. Sci. Technol.* 30:1120-1128.

Lollar, B. S., G. F. Slater, B. Sleep, M. Witt, G. M. Klecka, M. Harkness and J. Spivack. 2001. Stable carbon isotope evidence for intrinsic bioremediation of tetrachloroethene and trichlorethene at Area 6, Dover Air Force Base. *Environ. Sci. Technol.* 35:261-269.

Meier-Augenstein W. 1995. On line recording of $^{13}\text{C}/^{12}\text{C}$ ratios and mass spectra in one gas chromatographic analysis. *High Resol Chromatogr* 18, 28-32.

O'Malley V. P., T. A. Abrajano Jr., J. Hellou. 1994. Determination of the $^{13}\text{C}/^{12}\text{C}$ ratios of individual PAH from environmental samples: Can PAH sources be apportioned? *Org Geochem* 21:809-822.

O'Malley V. P., T. A. Abrajano Jr., J. Hellou. 1996. Stable carbon isotopic apportionment of individual polycyclic aromatic hydrocarbons in St. John's Harbour, Newfoundland. *Environ Sci Technol* 30:634-639.

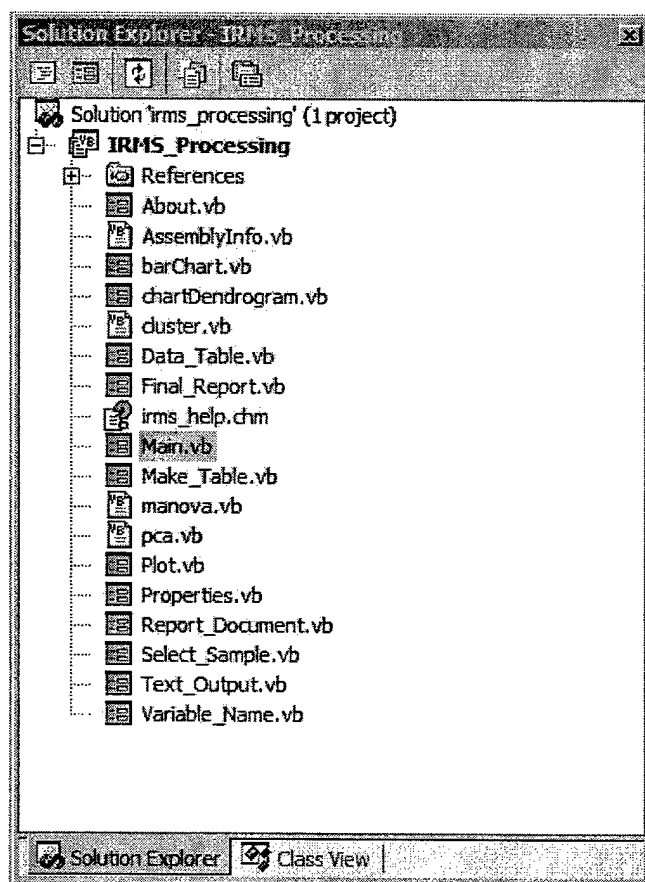
Peterson BJ, Fry B. 1987. Stable isotopes in ecosystem studies. *Ann Rev Ecol Syst* 18:293-320.

Silfer JA, Engel MH, Macko SA, Jumeau EJ. 1991. Stable carbon isotope analysis of amino acid enantiomers by conventional isotope ratio mass spectrometry and combined gas chromatography/isotope ratio mass spectrometry. *Anal Chem* 63, 370-374.

Trust BA, Mueller JG, Coffin RB, Cifuentes LA. 1995. The biodegradation of fluoranthene as monitored using stable carbon isotopes. Battelle In Situ and On-Site Bioreclamation Program Publication, San Diego CA, April 24-27, 1995 3:233-233.

APPENDIX I

Code listing. The following pages list all of the Visual Basic Code used in the application. The individual component objects used in the project can be seen in the figure below. Code for each is included.



```
Imports Cl.Win.ClFlexGrid
Imports System.Text.RegularExpressions
Imports System.Data
Imports System.Drawing
Imports System.ComponentModel
Imports System.Collections
Imports Cl.Common
Imports System.Drawing.Imaging
Imports System.Math
Imports System.Drawing.Printing
Imports System.IO
Imports Cl.ClPrintDocument
```

```
Public Class Form1
    Inherits System.Windows.Forms.Form
```

```
#Region " Windows Form Designer generated code "
```

```
    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
```

```
End Sub
```

```
    'Form overrides dispose to clean up the component list.
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub
```

```
    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
```

```
    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents MainMenu1 As System.Windows.Forms.MainMenu
    Friend WithEvents MenuItem6 As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem10 As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem12 As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem18 As System.Windows.Forms.MenuItem
    Friend WithEvents PrintPreviewDialog1 As System.Windows.Forms.PrintPreviewDialog
    Friend WithEvents mnuFileExit As System.Windows.Forms.MenuItem
    Friend WithEvents mnuFile As System.Windows.Forms.MenuItem
    Friend WithEvents mnuFileNew As System.Windows.Forms.MenuItem
    Friend WithEvents mnuFileOpen As System.Windows.Forms.MenuItem
    Friend WithEvents mnuFileSave As System.Windows.Forms.MenuItem
    Friend WithEvents mnuFileSaveAs As System.Windows.Forms.MenuItem
    Friend WithEvents FilePrintPreview As System.Windows.Forms.MenuItem
    Friend WithEvents mnuFilePrint As System.Windows.Forms.MenuItem
    Friend WithEvents mnuFileProperties As System.Windows.Forms.MenuItem
    Friend WithEvents mnuEdit As System.Windows.Forms.MenuItem
    Friend WithEvents mnuEditUndo As System.Windows.Forms.MenuItem
    Friend WithEvents mnuEditCut As System.Windows.Forms.MenuItem
    Friend WithEvents mnuEditCopy As System.Windows.Forms.MenuItem
    Friend WithEvents mnuEditPaste As System.Windows.Forms.MenuItem
    Friend WithEvents mnuEditDelete As System.Windows.Forms.MenuItem
```

```

Friend WithEvents mnuDeleteTable As System.Windows.Forms.MenuItem
Friend WithEvents mnuInsert As System.Windows.Forms.MenuItem
Friend WithEvents mnuInsertColumns As System.Windows.Forms.MenuItem
Friend WithEvents mnuFormat As System.Windows.Forms.MenuItem
Friend WithEvents mnuData As System.Windows.Forms.MenuItem
Friend WithEvents mnuDataStatistics As System.Windows.Forms.MenuItem
Friend WithEvents mnuWindow As System.Windows.Forms.MenuItem
Friend WithEvents mnuHelp As System.Windows.Forms.MenuItem
Friend WithEvents mnuHelpProgramHelp As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem2 As System.Windows.Forms.MenuItem
Friend WithEvents mnuHelpAbout As System.Windows.Forms.MenuItem
Friend WithEvents OpenFileDialog1 As System.Windows.Forms.OpenFileDialog
Friend WithEvents SaveFileDialog1 As System.Windows.Forms.SaveFileDialog
Friend WithEvents PrintDialog1 As System.Windows.Forms.PrintDialog
Friend WithEvents PageSetupDialog1 As System.Windows.Forms.PageSetupDialog
Friend WithEvents mnuDataProcess As System.Windows.Forms.MenuItem
Friend WithEvents mnuDataProcessManova As System.Windows.Forms.MenuItem
Friend WithEvents mnuDataProcessPCA As System.Windows.Forms.MenuItem
Friend WithEvents mnuDataProcessCluster As System.Windows.Forms.MenuItem
Friend WithEvents mnuWindowTile As System.Windows.Forms.MenuItem
Friend WithEvents mnuWindowCascade As System.Windows.Forms.MenuItem
Friend WithEvents mnuArrangeIcons As System.Windows.Forms.MenuItem
Friend WithEvents WindowCloseAll As System.Windows.Forms.MenuItem
Friend WithEvents mnuFormatCellsFont As System.Windows.Forms.MenuItem
Friend WithEvents mnuFormatCellsColor As System.Windows.Forms.MenuItem
Friend WithEvents mnuFileClose As System.Windows.Forms.MenuItem
Friend WithEvents Toolbar1 As System.Windows.Forms.ToolBar
Friend WithEvents tlbFileNew As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFileOpen As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFileClose As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFileSave As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbEditCut As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbEditCopy As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbEditPaste As System.Windows.Forms.ToolBarButton
Friend WithEvents ImageList1 As System.Windows.Forms.ImageList
Friend WithEvents ToolbarButton1 As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFilePrint As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton2 As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFormatBold As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFormatItalics As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFormatUnderline As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFormatLeftJustified As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFormatCenterJustified As System.Windows.Forms.ToolBarButton
Friend WithEvents tlbFormatRightJustified As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton3 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton4 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton5 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton6 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton7 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton8 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton13 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton14 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton15 As System.Windows.Forms.ToolBarButton
Friend WithEvents ToolbarButton16 As System.Windows.Forms.ToolBarButton
Friend WithEvents mnuInsertRows As System.Windows.Forms.MenuItem
Friend WithEvents mnuFormatCells As System.Windows.Forms.MenuItem
Friend WithEvents FontDialog1 As System.Windows.Forms.FontDialog
Friend WithEvents ColorDialog1 As System.Windows.Forms.ColorDialog
Friend WithEvents mnuFormatChart As System.Windows.Forms.MenuItem
Friend WithEvents doc As Cl.C1PrintDocument.C1PrintDocument
Friend WithEvents HelpProvider1 As System.Windows.Forms.HelpProvider
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.components = New System.ComponentModel.Container
    Dim resources As System.Resources.ResourceManager = New System.Resources.
ResourceManager(GetType(Form1))
    Me.MainMenu1 = New System.Windows.Forms.MainMenu
    Me.mnuFile = New System.Windows.Forms.MenuItem

```

```
Me.mnuFileNew = New System.Windows.Forms.MenuItem
Me.mnuFileOpen = New System.Windows.Forms.MenuItem
Me.mnuFileClose = New System.Windows.Forms.MenuItem
Me.mnuFileSave = New System.Windows.Forms.MenuItem
Me.mnuFileSaveAs = New System.Windows.Forms.MenuItem
Me.MenuItem6 = New System.Windows.Forms.MenuItem
Me.FilePrintPreview = New System.Windows.Forms.MenuItem
Me.mnuFilePrint = New System.Windows.Forms.MenuItem
Me.MenuItem10 = New System.Windows.Forms.MenuItem
Me.mnuFileProperties = New System.Windows.Forms.MenuItem
Me.MenuItem12 = New System.Windows.Forms.MenuItem
Me.mnuFileExit = New System.Windows.Forms.MenuItem
Me.mnuEdit = New System.Windows.Forms.MenuItem
Me.mnuEditUndo = New System.Windows.Forms.MenuItem
Me.mnuEditCut = New System.Windows.Forms.MenuItem
Me.mnuEditCopy = New System.Windows.Forms.MenuItem
Me.mnuEditPaste = New System.Windows.Forms.MenuItem
Me.MenuItem18 = New System.Windows.Forms.MenuItem
Me.mnuEditDelete = New System.Windows.Forms.MenuItem
Me.mnuDeleteTable = New System.Windows.Forms.MenuItem
Me.mnuInsert = New System.Windows.Forms.MenuItem
Me.mnuInsertColumns = New System.Windows.Forms.MenuItem
Me.mnuInsertRows = New System.Windows.Forms.MenuItem
Me.mnuFormat = New System.Windows.Forms.MenuItem
Me.mnuFormatCells = New System.Windows.Forms.MenuItem
Me.mnuFormatCellsFont = New System.Windows.Forms.MenuItem
Me.mnuFormatCellsColor = New System.Windows.Forms.MenuItem
Me.mnuFormatChart = New System.Windows.Forms.MenuItem
Me.mnuData = New System.Windows.Forms.MenuItem
Me.mnuDataProcess = New System.Windows.Forms.MenuItem
Me.mnuDataProcessManova = New System.Windows.Forms.MenuItem
Me.mnuDataProcessPCA = New System.Windows.Forms.MenuItem
Me.mnuDataProcessCluster = New System.Windows.Forms.MenuItem
Me.mnuDataStatistics = New System.Windows.Forms.MenuItem
Me.mnuWindow = New System.Windows.Forms.MenuItem
Me.mnuWindowTile = New System.Windows.Forms.MenuItem
Me.mnuWindowCascade = New System.Windows.Forms.MenuItem
Me.mnuArrangeIcons = New System.Windows.Forms.MenuItem
Me.WindowCloseAll = New System.Windows.Forms.MenuItem
Me.mnuHelp = New System.Windows.Forms.MenuItem
Me.mnuHelpProgramHelp = New System.Windows.Forms.MenuItem
Me.MenuItem2 = New System.Windows.Forms.MenuItem
Me.mnuHelpAbout = New System.Windows.Forms.MenuItem
Me.PrintPreviewDialog1 = New System.Windows.Forms.PrintPreviewDialog
Me.OpenFileDialog1 = New System.Windows.Forms.OpenFileDialog
Me.SaveFileDialog1 = New System.Windows.Forms.SaveFileDialog
Me.PrintDialog1 = New System.Windows.Forms.PrintDialog
Me.PageSetupDialog1 = New System.Windows.Forms.PageSetupDialog
Me.ToolBar1 = New System.Windows.Forms.ToolBar
Me.tlbFileNew = New System.Windows.Forms.ToolBarButton
Me.tlbFileOpen = New System.Windows.Forms.ToolBarButton
Me.tlbFileClose = New System.Windows.Forms.ToolBarButton
Me.tlbFileSave = New System.Windows.Forms.ToolBarButton
Me.tlbFilePrint = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton1 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton4 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton3 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton5 = New System.Windows.Forms.ToolBarButton
Me.tlbEditCut = New System.Windows.Forms.ToolBarButton
Me.tlbEditCopy = New System.Windows.Forms.ToolBarButton
Me.tlbEditPaste = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton2 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton6 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton7 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton8 = New System.Windows.Forms.ToolBarButton
Me.tlbFormatBold = New System.Windows.Forms.ToolBarButton
Me.tlbFormatItalics = New System.Windows.Forms.ToolBarButton
```

```

Me.tlbFormatUnderline = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton13 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton14 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton15 = New System.Windows.Forms.ToolBarButton
Me.ToolBarButton16 = New System.Windows.Forms.ToolBarButton
Me.tlbFormatLeftJustified = New System.Windows.Forms.ToolBarButton
Me.tlbFormatCenterJustified = New System.Windows.Forms.ToolBarButton
Me.tlbFormatRightJustified = New System.Windows.Forms.ToolBarButton
Me.ImageList1 = New System.Windows.Forms.ImageList(Me.components)
Me.FontDialog1 = New System.Windows.Forms.FontDialog
Me.ColorDialog1 = New System.Windows.Forms.ColorDialog
Me.doc = New Cl.C1PrintDocument.C1PrintDocument
Me.HelpProvider1 = New System.Windows.Forms.HelpProvider
Me.SuspendLayout()
'
'MainMenu1
'
Me.MainMenu1.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.mnuFile, ✓
Me.mnuEdit, Me.mnuInsert, Me.mnuFormat, Me.mnuData, Me.mnuWindow, Me.mnuHelp})
'
'mnuFile
'
Me.mnuFile.Index = 0
Me.mnuFile.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.mnuFileNew, ✓
Me.mnuFileOpen, Me.mnuFileClose, Me.mnuFileSave, Me.mnuFileSaveAs, Me.MenuItem6, Me. ✓
FilePrintPreview, Me.mnuFilePrint, Me.MenuItem10, Me.mnuFileProperties, Me.MenuItem12, ✓
Me.mnuFileExit})
Me.mnuFile.Text = "&File"
'
'mnuFileNew
'
Me.mnuFileNew.Index = 0
Me.mnuFileNew.Shortcut = System.Windows.Forms.Shortcut.CtrlN
Me.mnuFileNew.Text = "&New"
'
'mnuFileOpen
'
Me.mnuFileOpen.Index = 1
Me.mnuFileOpen.Shortcut = System.Windows.Forms.Shortcut.CtrlO
Me.mnuFileOpen.Text = "&Open"
'
'mnuFileClose
'
Me.mnuFileClose.Index = 2
Me.mnuFileClose.Text = "&Close"
'
'mnuFileSave
'
Me.mnuFileSave.Index = 3
Me.mnuFileSave.Shortcut = System.Windows.Forms.Shortcut.CtrlS
Me.mnuFileSave.Text = "&Save"
'
'mnuFileSaveAs
'
Me.mnuFileSaveAs.Index = 4
Me.mnuFileSaveAs.Text = "Save &As..."
'
'MenuItem6
'
Me.MenuItem6.Index = 5
Me.MenuItem6.Text = "- "
'
'FilePrintPreview
'
Me.FilePrintPreview.Index = 6
Me.FilePrintPreview.Text = "Print Pre&view"
'

```

```

        'mnuFilePrint
        '
        Me.mnuFilePrint.Index = 7
        Me.mnuFilePrint.Text = "&Print"
        '
        'MenuItem10
        '
        Me.MenuItem10.Index = 8
        Me.MenuItem10.Text = "- "
        '
        'mnuFileProperties
        '
        Me.mnuFileProperties.Index = 9
        Me.mnuFileProperties.Text = "Propert&ies"
        '
        'MenuItem12
        '
        Me.MenuItem12.Index = 10
        Me.MenuItem12.Text = "- "
        '
        'mnuFileExit
        '
        Me.mnuFileExit.Index = 11
        Me.mnuFileExit.Shortcut = System.Windows.Forms.Shortcut.AltF4
        Me.mnuFileExit.Text = "E&xit"
        '
        'mnuEdit
        '
        Me.mnuEdit.Index = 1
        Me.mnuEdit.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.mnuEditUndo, Me.mnuEditCut, Me.mnuEditCopy, Me.mnuEditPaste, Me.MenuItem18, Me.mnuEditDelete, Me.mnuDeleteTable})
        Me.mnuEdit.Text = "&Edit"
        '
        'mnuEditUndo
        '
        Me.mnuEditUndo.Enabled = False
        Me.mnuEditUndo.Index = 0
        Me.mnuEditUndo.Text = "&Undo"
        '
        'mnuEditCut
        '
        Me.mnuEditCut.Index = 1
        Me.mnuEditCut.Text = "Cu&t"
        '
        'mnuEditCopy
        '
        Me.mnuEditCopy.Index = 2
        Me.mnuEditCopy.Text = "&Copy"
        '
        'mnuEditPaste
        '
        Me.mnuEditPaste.Index = 3
        Me.mnuEditPaste.Text = "&Paste"
        '
        'MenuItem18
        '
        Me.MenuItem18.Index = 4
        Me.MenuItem18.Text = "- "
        '
        'mnuEditDelete
        '
        Me.mnuEditDelete.Index = 5
        Me.mnuEditDelete.Text = "&Delete"
        '
        'mnuDeleteTable
        '

```

```

Me.mnuDeleteTable.Index = 6
Me.mnuDeleteTable.Text = "De&lete Column(s)"
'
'mnuInsert
'
Me.mnuInsert.Index = 2
Me.mnuInsert.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
mnuInsertColumns, Me.mnuInsertRows})
Me.mnuInsert.Text = "&Insert"
Me.mnuInsert.Visible = False
'
'mnuInsertColumns
'
Me.mnuInsertColumns.Index = 0
Me.mnuInsertColumns.Text = "&Columns"
Me.mnuInsertColumns.Visible = False
'
'mnuInsertRows
'
Me.mnuInsertRows.Enabled = False
Me.mnuInsertRows.Index = 1
Me.mnuInsertRows.Text = "&Rows"
'
'mnuFormat
'
Me.mnuFormat.Index = 3
Me.mnuFormat.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
mnuFormatCells, Me.mnuFormatChart})
Me.mnuFormat.Text = "F&ormat"
'
'mnuFormatCells
'
Me.mnuFormatCells.Index = 0
Me.mnuFormatCells.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
mnuFormatCellsFont, Me.mnuFormatCellsColor})
Me.mnuFormatCells.Text = "&Cells"
'
'mnuFormatCellsFont
'
Me.mnuFormatCellsFont.Index = 0
Me.mnuFormatCellsFont.Text = "&Font"
'
'mnuFormatCellsColor
'
Me.mnuFormatCellsColor.Index = 1
Me.mnuFormatCellsColor.Text = "&Color"
'
'mnuFormatChart
'
Me.mnuFormatChart.Index = 1
Me.mnuFormatChart.Text = "C&hart"
'
'mnuData
'
Me.mnuData.Index = 4
Me.mnuData.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
mnuDataProcess, Me.mnuDataStatistics})
Me.mnuData.Text = "&Data"
'
'mnuDataProcess
'
Me.mnuDataProcess.Index = 0
Me.mnuDataProcess.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
mnuDataProcessManova, Me.mnuDataProcessPCA, Me.mnuDataProcessCluster})
Me.mnuDataProcess.Text = "&Process"
'
'mnuDataProcessManova

```

```

    ,
    Me.mnuDataProcessManova.Index = 0
    Me.mnuDataProcessManova.Text = "&Manova"
    ,
    'mnuDataProcessPCA
    ,
    Me.mnuDataProcessPCA.Index = 1
    Me.mnuDataProcessPCA.Text = "&PCA"
    ,
    'mnuDataProcessCluster
    ,
    Me.mnuDataProcessCluster.Index = 2
    Me.mnuDataProcessCluster.Text = "&Cluster"
    ,
    'mnuDataStatistics
    ,
    Me.mnuDataStatistics.Index = 1
    Me.mnuDataStatistics.Text = "S&tatistics"
    Me.mnuDataStatistics.Visible = False
    ,
    'mnuWindow
    ,
    Me.mnuWindow.Index = 5
    Me.mnuWindow.MdiList = True
    Me.mnuWindow.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
mnuWindowTile, Me.mnuWindowCascade, Me.mnuArrangeIcons, Me.WindowCloseAll})
    Me.mnuWindow.Text = "&Window"
    ,
    'mnuWindowTile
    ,
    Me.mnuWindowTile.Index = 0
    Me.mnuWindowTile.Text = "&Tile"
    ,
    'mnuWindowCascade
    ,
    Me.mnuWindowCascade.Index = 1
    Me.mnuWindowCascade.Text = "&Cascade"
    ,
    'mnuArrangeIcons
    ,
    Me.mnuArrangeIcons.Index = 2
    Me.mnuArrangeIcons.Text = "&Arrange Icons"
    ,
    'WindowCloseAll
    ,
    Me.WindowCloseAll.Index = 3
    Me.WindowCloseAll.Text = "&Close All"
    ,
    'mnuHelp
    ,
    Me.mnuHelp.Index = 6
    Me.mnuHelp.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
mnuHelpProgramHelp, Me.MenuItem2, Me.mnuHelpAbout})
    Me.mnuHelp.Text = "&Help"
    ,
    'mnuHelpProgramHelp
    ,
    Me.mnuHelpProgramHelp.Index = 0
    Me.mnuHelpProgramHelp.Text = "&Program Help"
    ,
    'MenuItem2
    ,
    Me.MenuItem2.Index = 1
    Me.MenuItem2.Text = "-"
    ,
    'mnuHelpAbout
    ,

```

```

Me.mnuHelpAbout.Index = 2
Me.mnuHelpAbout.Text = "&About..."
'
'PrintPreviewDialog1
'
Me.PrintPreviewDialog1.AutoScrollMargin = New System.Drawing.Size(0, 0)
Me.PrintPreviewDialog1.AutoScrollMinSize = New System.Drawing.Size(0, 0)
Me.PrintPreviewDialog1.ClientSize = New System.Drawing.Size(400, 300)
Me.PrintPreviewDialog1.Enabled = True
Me.PrintPreviewDialog1.Icon = CType(resources.GetObject("PrintPreviewDialog1.Icon"), System.Drawing.Icon)
Me.PrintPreviewDialog1.Location = New System.Drawing.Point(125, 15)
Me.PrintPreviewDialog1.MinimumSize = New System.Drawing.Size(375, 250)
Me.PrintPreviewDialog1.Name = "PrintPreviewDialog1"
Me.PrintPreviewDialog1.TransparencyKey = System.Drawing.Color.Empty
Me.PrintPreviewDialog1.Visible = False
'
'ToolBar1
'
Me.ToolBar1.Buttons.AddRange(New System.Windows.Forms.ToolBarButton() {Me.
tlbFileNew, Me.tlbFileOpen, Me.tlbFileClose, Me.tlbFileSave, Me.tlbFilePrint, Me.
ToolBarButton1, Me.ToolBarButton4, Me.ToolBarButton3, Me.ToolBarButton5, Me.tlbEditCut,
Me.tlbEditCopy, Me.tlbEditPaste, Me.ToolBarButton2, Me.ToolBarButton6, Me.
ToolBarButton7, Me.ToolBarButton8, Me.tlbFormatBold, Me.tlbFormatItalics, Me.
tlbFormatUnderline, Me.ToolBarButton13, Me.ToolBarButton14, Me.ToolBarButton15, Me.
ToolBarButton16, Me.tlbFormatLeftJustified, Me.tlbFormatCenterJustified, Me.
tlbFormatRightJustified})
Me.ToolBar1.DropDownArrows = True
Me.ToolBar1.ImageList = Me.ImageList1
Me.ToolBar1.Location = New System.Drawing.Point(0, 0)
Me.ToolBar1.Name = "ToolBar1"
Me.ToolBar1.ShowToolTips = True
Me.ToolBar1.Size = New System.Drawing.Size(974, 28)
Me.ToolBar1.TabIndex = 1
'
'tlbFileNew
'
Me.tlbFileNew.ImageIndex = 14
Me.tlbFileNew.ToolTipText = "New File"
'
'tlbFileOpen
'
Me.tlbFileOpen.ImageIndex = 16
Me.tlbFileOpen.ToolTipText = "Open File"
'
'tlbFileClose
'
Me.tlbFileClose.ImageIndex = 2
Me.tlbFileClose.ToolTipText = "Close File"
'
'tlbFileSave
'
Me.tlbFileSave.ImageIndex = 23
Me.tlbFileSave.ToolTipText = "Save"
'
'tlbFilePrint
'
Me.tlbFilePrint.ImageIndex = 21
Me.tlbFilePrint.ToolTipText = "Print"
'
'ToolBarButton1
'
Me.ToolBarButton1.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
'
'ToolBarButton4
'
Me.ToolBarButton4.Style = System.Windows.Forms.ToolBarButtonStyle.Separator

```

```

    ,
    'ToolBarButton3
    ,
    Me.ToolBarButton3.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,
    'ToolBarButton5
    ,
    Me.ToolBarButton5.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,
    'tlbEditCut
    ,
    Me.tlbEditCut.ImageIndex = 4
    Me.tlbEditCut.ToolTipText = "Cut"
    ,
    'tlbEditCopy
    ,
    Me.tlbEditCopy.ImageIndex = 3
    Me.tlbEditCopy.ToolTipText = "Copy"
    ,
    'tlbEditPaste
    ,
    Me.tlbEditPaste.ImageIndex = 20
    Me.tlbEditPaste.ToolTipText = "Paste"
    ,
    'ToolBarButton2
    ,
    Me.ToolBarButton2.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,
    'ToolBarButton6
    ,
    Me.ToolBarButton6.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,
    'ToolBarButton7
    ,
    Me.ToolBarButton7.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,
    'ToolBarButton8
    ,
    Me.ToolBarButton8.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,
    'tlbFormatBold
    ,
    Me.tlbFormatBold.ImageIndex = 0
    Me.tlbFormatBold.ToolTipText = "Bold"
    ,
    'tlbFormatItalics
    ,
    Me.tlbFormatItalics.ImageIndex = 6
    Me.tlbFormatItalics.ToolTipText = "Italics"
    ,
    'tlbFormatUnderline
    ,
    Me.tlbFormatUnderline.ImageIndex = 27
    Me.tlbFormatUnderline.ToolTipText = "Underline"
    ,
    'ToolBarButton13
    ,
    Me.ToolBarButton13.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,
    'ToolBarButton14
    ,
    Me.ToolBarButton14.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,
    'ToolBarButton15
    ,
    Me.ToolBarButton15.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
    ,

```

```

'ToolBarButton16
'
Me.ToolBarButton16.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
'
'tlbFormatLeftJustified
'
Me.tlbFormatLeftJustified.ImageIndex = 9
Me.tlbFormatLeftJustified.ToolTipText = "Left Justified"
'
'tlbFormatCenterJustified
'
Me.tlbFormatCenterJustified.ImageIndex = 8
Me.tlbFormatCenterJustified.ToolTipText = "Center Justified"
'
'tlbFormatRightJustified
'
Me.tlbFormatRightJustified.ImageIndex = 11
Me.tlbFormatRightJustified.ToolTipText = "Right Justified"
'
'ImageList1
'
Me.ImageList1.ImageSize = New System.Drawing.Size(16, 16)
Me.ImageList1.ImageStream = CType(resources.GetObject("ImageList1.ImageStream"),
System.Windows.Forms.ImageListStreamer)
Me.ImageList1.TransparentColor = System.Drawing.Color.Transparent
'
'doc
'
Me.doc.ClDPageSettings = "color:True;landscape:False;margins:100,100,100,100;
papersize:850,1100,TABLAHQAdAB" & _
"1AHIAIAAoADgALgAlACAAeAAgADEAMQAgAGkAbgAuACKAIAA="
Me.doc.ColumnSpacingStr = "0.5in"
Me.doc.ColumnSpacingUnit.DefaultType = True
Me.doc.ColumnSpacingUnit.UnitValue = "0.5in"
Me.doc.DefaultUnit = Cl.ClPrintDocument.UnitTypeEnum.Inch
Me.doc.DocumentName = ""
'
'HelpProvider1
'
Me.HelpProvider1.HelpNamespace = "C:\Documents and Settings\tjb\My Documents\
Visual Studio Projects\IRMS_Processing" & _
" _NC\Help\IRMS_help.chm"
'
'Form1
'
Me.AccessibleDescription = ""
Me.AccessibleName = ""
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.ClientSize = New System.Drawing.Size(974, 753)
Me.Controls.Add(Me.ToolBar1)
Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
Me.IsMdiContainer = True
Me.Menu = Me.MainMenu1
Me.Name = "Form1"
Me.Text = "IRMS Data Processing"
Me.ResumeLayout(False)

```

End Sub

#End Region

```

Private myColumnsOfData As Integer
Private myRowsOfData As Integer
Private lastFilterIndex As Integer = 1
Private myManova As manovaexpandtjb.expandtable
Private myManova_p As manova_probability.manova_p

```

```

Private myPCA_Output As PCA_output.PCA_output_data
Private myLinkages As linkages.pdist_linkage
Private myClusterLinks As clusters_tjb.clusterlinks
Private myFileName As String
Private myDendrogram As dendrogram_tjb.dendrogram_tjb_output
Private myManova_stats As manova_stats.manova_stats
Private myManova_stats_expanded As manova_expand_stats.manova_expand_statistics
Private myManova_statsFunctions As manova_expand_stats.manova_expand_statistics
Friend WithEvents currentGrid As Cl.Win.ClFlexGrid.ClFlexGrid

Private Property RowsOfData() As Integer
    Get
        Return myRowsOfData
    End Get
    Set(ByVal Value As Integer)
        myRowsOfData = Value
    End Set
End Property

Private Property ColumnsOfData() As Integer
    Get
        Return myColumnsOfData
    End Get
    Set(ByVal Value As Integer)
        myColumnsOfData = Value
    End Set
End Property

Private ReadOnly Property FileName() As String
    Get
        Return CType(myFileName, String)
    End Get
End Property

Dim WithEvents PrintDoc As PrintDocument
Dim currPage As Integer
Dim lastPage As Integer

Dim myActiveForm As Form

Public Sub ShowGrid(ByVal grid As Cl.Win.ClFlexGrid.ClFlexGrid)
    currentGrid = grid
    MakeDoc(Me.doc, Nothing)
    Dim aprev As New Final_Report
    AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf MakeDoc)
    aprev.ClPrintPreview1.Document = Me.doc
    aprev.ShowDialog()
    RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf MakeDoc)
    currentGrid = Nothing
    aprev.Dispose()
End Sub

Private Sub MakeDoc(ByVal doc As ClPrintDocument, ByVal e As GenerateEventArgs)
    Dim StyleText As New ClDocStyle(doc)
    StyleText.ShapeLine = New LineDef(Color.White, 1)
    StyleText.ShapeFillColor = Color.Transparent
    StyleText.TextAlignHorz = Cl.ClPrintDocument.AlignHorzEnum.Center
    StyleText.Font = New Font("Times New Roman", 14, FontStyle.Bold)
    StyleText.TextColor = Color.Black
    With doc
        .DefaultUnit = Cl.ClPrintDocument.UnitTypeEnum.Mm
        .PageHeader.Height = 0
        .PageFooter.Height = 5
        .PageHeader.RenderText.Style.TextAlignHorz = Cl.ClPrintDocument.AlignHorzEnum.
    End With
    .Justify

```

```

        '.PageHeader.RenderText.Text = "Page [@@PageNo@@] of [@@PageCount@@]"
        .PageFooter.RenderText.Style.TextAlignHorz = C1.C1PrintDocument.AlignHorzEnum.✓
Justify
        .PageFooter.RenderText.Style.TextAlignVert = C1.C1PrintDocument.AlignVertEnum.✓
Bottom
        .PageFooter.RenderText.Text = "Page [@@PageNo@@] of [@@PageCount@@]"
        .StartDoc()

        .RenderBlockC1Printable(currentGrid, .BodyAreaSize.Width, .BodyAreaSize.Height✓
, Nothing, StyleText)
        .EndDoc()
    End With
End Sub

```

```
Public Sub InternalPrintGrid(ByVal flexgrid As C1FlexGrid)
```

```

    ' get grid's PrintDocument object
    Dim pd As System.Drawing.Printing.PrintDocument
    pd = flexgrid.PrintParameters.PrintDocument()

    ' set up the page (landscape, 1.5" left margin)
    With pd.DefaultPageSettings
        .Landscape = True
        .Margins.Left = 150
    End With

    ' set up header and footer fonts
    flexgrid.PrintParameters.HeaderFont = New Font("Arial Black", 14, FontStyle.Bold)
    flexgrid.PrintParameters.FooterFont = New Font("Arial Narrow", 8, FontStyle.Italic✓
)

    ' preview the grid
    flexgrid.PrintGrid(flexgrid.Text, PrintGridFlags.ShowPreviewDialog)
    'flexgrid.PrintGrid("VB Tutorial", PrintGridFlags.ShowPreviewDialog, _
    '    "VB Tutorial" + Chr(9) + Chr(9) + Format(DateTime.Now, "d"), _
    '    Chr(9) + Chr(9) + "Page {0} of {1}")
End Sub

```

```
Public Sub CheckForExistingInstance()
```

```

    'Get number of processes of you program
    If Process.GetProcessesByName _
        (Process.GetCurrentProcess.ProcessName).Length > 1 Then

        MessageBox.Show _
            ("Another Instance of this process is already running", _
            "Multiple Instances Forbidden", _
            MessageBoxButtons.OK, _
            MessageBoxIcon.Exclamation)
        Application.Exit()
    End If
End Sub

```

```
Private Sub mnuFileOpen_Click(ByVal sender As System.Object, ByVal e As System.✓
EventArgs) Handles mnuFileOpen.Click
```

```

    Dim OpenDlg As New OpenFileDialog
    Dim DataTable As New Data_Table
    'DataTable = CType(Me.ActiveMdiChild, Data_Table)
    With OpenDlg
        .FileName = ""
        .Filter = "Comma Separated (*.csv)|*.csv|Text files (*.txt)|*.txt|All files (*✓
.*)|*.*"
        .FilterIndex = 1
        .CheckFileExists = True
    End With

```

```

    If .ShowDialog() = DialogResult.Cancel Then Return
    Try
        If OpenFileDialog.FileName.EndsWith(".csv") Then
            DataTable.DataTable.LoadGrid(OpenDlg.FileName, FileFormatEnum.
TextComma, True)
            DataTable.MdiParent = Me
            DataTable.Show()
        End If
        If OpenFileDialog.FileName.EndsWith(".txt") Then
            Dim txtOutput As New Text_Output
            txtOutput.dataReport.LoadFile(OpenDlg.FileName)
            txtOutput.MdiParent = Me
            txtOutput.Show()
        End If

        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error Loading File", MessageBoxButtons.
OKCancel, MessageBoxIcon.Hand)
        End Try
    End With

End Sub

Private Sub mnuFileSaveAs_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFileSaveAs.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If

    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim SaveAsDlg As New SaveFileDialog
        Dim DataTable As New Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        With SaveAsDlg
            .FileName = ""
            .Filter = "Comma Separated (*.csv)|*.csv|All files (*.*)|*.*"
            .FilterIndex = 1
            If .ShowDialog() = DialogResult.Cancel Then Return
            DataTable.DataTable.SaveGrid(SaveAsDlg.FileName, FileFormatEnum.TextComma,
True)
        End With
        myFileName = SaveAsDlg.FileName
    End If

    If Me.ActiveMdiChild.Name = "Plot" Then
        lastFilterIndex = 1
        Dim myPlot As Plot = CType(Me.ActiveMdiChild, Plot)
        Dim sfg As New SaveFileDialog

        sfg.Filter = "Metafiles (*.emf)|*.emf|" + "Bmp files (*.bmp)|*.bmp|" + "Gif
files (*.gif)|*.gif|" + "Jpeg files (*.jpg;*.jpeg)|*.jpg;*.jpeg|" + "Png files (*.png)
|*.png|" + "All graphic files (*.emf;*.bmp;*.gif;*.jpg;*.jpeg;*.png)|*.emf;*.bmp;*.gif
;*.jpg;*.jpeg;*.png"
        sfg.FilterIndex = lastFilterIndex
        sfg.OverwritePrompt = True
        sfg.CheckPathExists = True
        sfg.RestoreDirectory = False
        sfg.ValidateNames = True

        If sfg.ShowDialog() = DialogResult.OK Then
            Dim fn As String = sfg.FileName
            Dim index As Integer = fn.LastIndexOf(".")
            If index < 0 Then
                index = fn.Length + 1
                fn += ".emf"
            Else

```

```

        indext += 1
    End If
    Dim ext As String = fn.Substring(indext)
    Dim imgfmt As ImageFormat = Nothing

    Select Case ext
        Case "emf"
            imgfmt = ImageFormat.Emf
            myPlot.chartPCA.SaveImage(fn, imgfmt)

        Case "bmp"
            imgfmt = ImageFormat.Bmp

        Case "gif"
            imgfmt = ImageFormat.Gif

        Case "jpeg", "jpg"
            imgfmt = ImageFormat.Jpeg

        Case "png"
            imgfmt = ImageFormat.Png

        Case Else
            Return
    End Select

    lastFilterIndex = sfg.FilterIndex

    If Not imgfmt.Equals(ImageFormat.Emf) Then
        Dim img As Image = myPlot.chartPCA.GetImage()
        img.Save(fn, imgfmt)
        img.Dispose()
    End If
End If
sfg.Dispose()
End If

If Me.ActiveMdiChild.Name = "barChart" Then
    lastFilterIndex = 1
    Dim mybarChart As barChart = CType(Me.ActiveMdiChild, barChart)
    Dim sfg As New SaveFileDialog

    sfg.Filter = "Metafiles (*.emf)|*.emf|" + "Bmp files (*.bmp)|*.bmp|" + "Gif
files (*.gif)|*.gif|" + "Jpeg files (*.jpg;*.jpeg)|*.jpg;*.jpeg|" + "Png files (*.png)
|*.png|" + "All graphic files (*.emf;*.bmp;*.gif;*.jpg;*.jpeg;*.png)|*.emf;*.bmp;*.gif
;*.jpg;*.jpeg;*.png"
    sfg.FilterIndex = lastFilterIndex
    sfg.OverwritePrompt = True
    sfg.CheckPathExists = True
    sfg.RestoreDirectory = False
    sfg.ValidateNames = True

    If sfg.ShowDialog() = DialogResult.OK Then
        Dim fn As String = sfg.FileName
        Dim indext As Integer = fn.LastIndexOf(".")
        If indext < 0 Then
            indext = fn.Length + 1
            fn += ".emf"
        Else
            indext += 1
        End If
        Dim ext As String = fn.Substring(indext)
        Dim imgfmt As ImageFormat = Nothing

        Select Case ext
            Case "emf"
                imgfmt = ImageFormat.Emf

```

```

        mybarChart.chartBar.SaveImage(fn, imgfmt)

        Case "bmp"
            imgfmt = ImageFormat.Bmp

        Case "gif"
            imgfmt = ImageFormat.Gif

        Case "jpeg", "jpg"
            imgfmt = ImageFormat.Jpeg

        Case "png"
            imgfmt = ImageFormat.Png

        Case Else
            Return
        End Select

        lastFilterIndex = sfg.FilterIndex

        If Not imgfmt.Equals(ImageFormat.Emf) Then
            Dim img As Image = mybarChart.chartBar.GetImage()
            img.Save(fn, imgfmt)
            img.Dispose()
        End If
        sfg.Dispose()
    End If

    If Me.ActiveMdiChild.Name = "chartDendrogram" Then
        lastFilterIndex = 1
        Dim mychartDendrogram As chartDendrogram = CType(Me.ActiveMdiChild,
chartDendrogram)
        Dim sfg As New SaveFileDialog

        sfg.Filter = "Metafiles (*.emf)|*.emf|" + "Bmp files (*.bmp)|*.bmp|" + "Gif
files (*.gif)|*.gif|" + "Jpeg files (*.jpg;*.jpeg)|*.jpg;*.jpeg|" + "Png files (*.png)
|*.png|" + "All graphic files (*.emf;*.bmp;*.gif;*.jpg;*.jpeg;*.png)|*.emf;*.bmp;*.gif
;*.jpg;*.jpeg;*.png"
        sfg.FilterIndex = lastFilterIndex
        sfg.OverwritePrompt = True
        sfg.CheckPathExists = True
        sfg.RestoreDirectory = False
        sfg.ValidateNames = True

        If sfg.ShowDialog() = DialogResult.OK Then
            Dim fn As String = sfg.FileName
            Dim indext As Integer = fn.LastIndexOf(".")
            If indext < 0 Then
                indext = fn.Length + 1
                fn += ".emf"
            Else
                indext += 1
            End If
            Dim ext As String = fn.Substring(indext)
            Dim imgfmt As ImageFormat = Nothing

            Select Case ext
                Case "emf"
                    imgfmt = ImageFormat.Emf
                    mychartDendrogram.chDendrogram.SaveImage(fn, imgfmt)

                Case "bmp"
                    imgfmt = ImageFormat.Bmp

                Case "gif"
                    imgfmt = ImageFormat.Gif

```

```

        Case "jpeg", "jpg"
            imgfmt = ImageFormat.Jpeg

        Case "png"
            imgfmt = ImageFormat.Png

        Case Else
            Return
    End Select

    lastFilterIndex = sfg.FilterIndex

    If Not imgfmt.Equals(ImageFormat.Emf) Then
        Dim img As Image = mychartDendrogram.chDendrogram.GetImage()
        img.Save(fn, imgfmt)
        img.Dispose()
    End If
    sfg.Dispose()
End If

If Me.ActiveMdiChild.Name = "Text_Output" Then
    Dim TextOutput As Text_Output = CType(Me.ActiveMdiChild, Text_Output)
    Dim saveFileDialog As New SaveFileDialog
    saveFileDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*"
    saveFileDialog.FilterIndex = 1
    saveFileDialog.FileName = saveFileDialog.FileName
    If saveFileDialog.ShowDialog() = DialogResult.OK Then
        TextOutput.dataReport.SaveFile(saveFileDialog.FileName)
    End If
End If

End Sub

Private Sub mnuFileExit_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFileExit.Click
    Application.Exit()
End Sub

Private Sub mnuHelpAbout_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuHelpAbout.Click
    Dim AboutDlg As New About
    myActiveForm = AboutDlg
    myActiveForm.ShowDialog()
End Sub

Private Sub mnuFileNew_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFileNew.Click
    Dim DataTableMake As New Make_Table
    If DataTableMake.ShowDialog = DialogResult.OK Then
        Dim Replicates As String = DataTableMake.txtReplicateNumber.Text
        Dim VariableNames As New Variable_Names
        VariableNames.ShowDialog()
        If VariableNames.DialogResult = DialogResult.OK Then
            'Get all of the data in grid
            Dim i As Integer
            i = VariableNames.VariableNames.Rows.Count - 1
            'Define a grid with all of the data
            Dim VariableNamesList As New Cl.Win.ClFlexGrid.CellRange
            VariableNamesList = VariableNames.VariableNames.GetCellRange(1, 0, i, 0)
            'Find out how many variables were put in the original list by seeking

```

```

        'out any alpha character followed by a newline character
        Dim q As Integer = 0
        'Find where the alpha next to \n characters are
        Dim re As New Regex("[a-zA-Z0-9]\x0D")
        Dim mc As MatchCollection = re.Matches(VariableNamesList.Clip)
        'Find out how many alpha next to \n characters there are
        q = mc.Count
        'reget the cell range based on this number (plus the 3 (more than 0) that
are added in the beginning
        VariableNamesList = VariableNames.VariableNames.GetCellRange(1, 0, q + 3,
0)

        'Replace the \n characters with
        Dim DataTable As New Data_Table
        DataTable.MdiParent = Me
        DataTable.ColumnHeaders = VariableNamesList.Clip
        DataTable.Replicates = Replicates
        Me.ColumnsOfData = CType(Replicates, Integer)
        DataTable.Show()

    Else
        Dim DataTable As New Data_Table
        DataTable.MdiParent = Me
        DataTable.Replicates = Replicates
        Me.ColumnsOfData = CType(Replicates, Integer)
        DataTable.Show()
    End If
Else
    Dim Replicates As String = DataTableMake.txtReplicateNumber.Text
    Dim DataTable As New Data_Table
    DataTable.MdiParent = Me
    DataTable.Replicates = Replicates
    Me.ColumnsOfData = CType(Replicates, Integer)
    DataTable.Show()
End If
End Sub

Private Sub mnuFileProperties_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFileProperties.Click
    Dim Properties As New Properties
    Properties.MdiParent = Me
    Properties.Show()
    If Properties.DialogResult = DialogResult.OK Or Properties.DialogResult =
DialogResult.Cancel Then
        Return
    End If
    Return
End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
End Sub

Private Sub mnuFormatCells_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFormatCells.Click
End Sub

Private Sub mnuFormatCellsFont_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFormatCellsFont.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        Dim SelectedFont As Font
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim Selection As CellRange

```

```

        Selection = DataTable.DataTable.Selection
        Dim FontDlg As New FontDialog
        If FontDlg.ShowDialog = DialogResult.OK Then
            SelectedFont = FontDlg.Font
            Selection.StyleNew.Font = SelectedFont
        End If
    End If
End Sub

Private Sub mnuHelpProgramHelp_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuHelpProgramHelp.Click
    Help.ShowHelp(Me, HelpProvider1.HelpNamespace)
End Sub

Private Sub mnuWindowTile_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuWindowTile.Click
    Me.LayoutMdi(System.Windows.Forms.MdiLayout.TileVertical)
End Sub

Private Sub mnuFileClose_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFileClose.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    Me.ActiveMdiChild.Close()
End Sub

Private Sub mnuEditCopy_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuEditCopy.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Plot" Then
        Dim myPlot As Plot = CType(Me.ActiveMdiChild, Plot)
        myPlot.chartPCA.SaveImage(ImageFormat.Emf)
    End If
    If Me.ActiveMdiChild.Name = "chartDendrogram" Then
        Dim myDendrogram As chartDendrogram = CType(Me.ActiveMdiChild, chartDendrogram
)
        myDendrogram.chDendrogram.SaveImage(ImageFormat.Emf)
    End If
    If Me.ActiveMdiChild.Name = "barChart" Then
        Dim mybarChart As barChart = CType(Me.ActiveMdiChild, barChart)
        mybarChart.chartBar.SaveImage(ImageFormat.Emf)
    End If
    If Me.ActiveMdiChild.Name = "Text_Output" Then
        Dim myTextOutput As Text_Output = CType(Me.ActiveMdiChild, Text_Output)
        Dim Selection As String = myTextOutput.dataReport.SelectedText
        Clipboard.SetDataObject(Selection)
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Clipboard.SetDataObject(DataTable.DataTable.Clip)
    End If
End Sub

Private Sub mnuEditPaste_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuEditPaste.Click
    If Me.ActiveMdiChild Is Nothing Then

```

```

        Return
    End If

    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim data As IDataObject = Clipboard.GetDataObject()
        If data.GetDataPresent(DataFormats.Text) Then
            ' there is, so paste it
            DataTable.DataTable.Select(DataTable.DataTable.Row, DataTable.DataTable.
Col, DataTable.DataTable.Rows.Count - 1, DataTable.DataTable.Cols.Count - 1, False)
            DataTable.DataTable.Clip = CType(data.GetData(DataFormats.Text), String)
            DataTable.DataTable.Select(DataTable.DataTable.Row, DataTable.DataTable.
Col)
        End If
    End If

End Sub

Private Sub mnuEditCut_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuEditCut.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If

    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Clipboard.SetDataObject(DataTable.DataTable.Clip)
        Dim selected As CellRange
        selected = DataTable.DataTable.Selection
        selected.Data = Nothing
    End If

End Sub

Private Sub mnuEditDelete_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuEditDelete.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim selected As CellRange
        selected = DataTable.DataTable.Selection
        selected.Data = Nothing
    End If
End Sub

Private Sub mnuDeleteTable_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuDeleteTable.Click
    Dim DataTable As Data_Table
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim selectedColumns As CellRange
        selectedColumns = DataTable.DataTable.Selection
        Dim selectedColumnLower As Integer = selectedColumns.c1
        Dim selectedColumnUpper As Integer = selectedColumns.c2
        Dim columnRange As ColumnCollection
        columnRange = DataTable.DataTable.Cols
        columnRange.DefaultSize = 70
        Dim columnCount As Integer
        For columnCount = selectedColumnLower To selectedColumnUpper

```

```

        columnRange.Remove(columnCount)
    Next
End If
End Sub

Private Sub WindowCloseAll_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles WindowCloseAll.Click
    Dim ChildWindows As Integer
    Dim MdiChildren As Integer
    ChildWindows = Me.MdiChildren.GetLength(0)
    For MdiChildren = 1 To ChildWindows
        Me.ActiveMdiChild.Close()
    Next

End Sub

Private Sub mnuArrangeIcons_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuArrangeIcons.Click

End Sub

Private Sub mnuWindowCascade_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuWindowCascade.Click
End Sub

Private Sub mnuInsertColumns_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuInsertColumns.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    Dim DataTable As Data_Table
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim selectedColumns As CellRange
        selectedColumns = DataTable.DataTable.Selection
        Dim selectedColumnLower As Integer = selectedColumns.c1
        Dim selectedColumnUpper As Integer = selectedColumns.c2
        Dim columnRange As ColumnCollection
        columnRange = DataTable.DataTable.Cols
        columnRange.DefaultSize = 70
        Dim columnCount As Integer
        For columnCount = selectedColumnLower To selectedColumnUpper
            columnRange.Insert(columnCount)
        Next
    End If
End Sub

#Region " Toolbars "

Private Sub ToolBar1_ButtonClick(ByVal sender As System.Object, ByVal e As System.
Windows.Forms.ToolBarButtonClickEventArgs) Handles ToolBar1.ButtonClick
    Select Case ToolBar1.Buttons.IndexOf(e.Button)
        Case 0
            'New
            Dim DataTableMake As New Make_Table
            If DataTableMake.ShowDialog = DialogResult.OK Then
                Dim Replicates As String = DataTableMake.txtReplicateNumber.Text
                Dim VariableNames As New Variable_Names
                VariableNames.ShowDialog()
                If VariableNames.DialogResult = DialogResult.OK Then
                    'Get all of the data in grid
                    Dim i As Integer
                    i = VariableNames.VariableNames.Rows.Count - 1
                    'Define a grid with all of the data
                    Dim VariableNamesList As New C1.Win.C1FlexGrid.CellRange
                    VariableNamesList = VariableNames.VariableNames.GetCellRange(1, 0,

```

```

i, 0)
seeking
    'Find out how many variables were put in the original list by
    'out any alpha character followed by a newline character
    Dim q As Integer = 0
    'Find where the alpha next to \n characters are
    Dim re As New Regex("[a-zA-Z]\x0D")
    Dim mc As MatchCollection = re.Matches(VariableNamesList.Clip)
    'Find out how many alpha next to \n characters there are
    q = mc.Count
    'reget the cell range based on this number (plus the 3 (more than
0) that are added in the beginning
    VariableNamesList = VariableNames.VariableNames.GetCellRange(1, 0,
q + 3, 0)

    'Replace the \n characters with
    Dim DataTable As New Data_Table
    DataTable.MdiParent = Me
    DataTable.ColumnHeaders = VariableNamesList.Clip
    DataTable.Replicates = Replicates
    DataTable.Show()

Else
    Dim DataTable As New Data_Table
    DataTable.MdiParent = Me
    DataTable.Replicates = Replicates
    DataTable.Show()
End If
Else
    Dim Replicates As String = DataTableMake.txtReplicateNumber.Text
    Dim DataTable As New Data_Table
    DataTable.MdiParent = Me
    DataTable.Replicates = Replicates
    DataTable.Show()
End If

Case 1
    'Open
    Dim OpenFileDialog As New OpenFileDialog
    With OpenFileDialog
        .FileName = ""
        .Filter = "Text files (*.txt)|*.txt|Comma Separated (*.csv)|*.xls|All
files (*.*)|*.*"
        .FilterIndex = 1
        .CheckFileExists = True
        If .ShowDialog() = DialogResult.Cancel Then Return
    End With

Case 2
    'Close
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    Me.ActiveMdiChild.Close()

Case 3
    'Save

Case 4
    'Print

Case 9
    'Cut
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Clipboard.SetDataObject(DataTable.DataTable.Clip)
        Dim selected As CellRange

```

```

        selected = DataTable.DataTable.Selection
        selected.Data = Nothing
    End If
Case 10
    'Copy
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If

    If Me.ActiveMdiChild.Name = "Plot" Then
        Dim myPlot As Plot = CType(Me.ActiveMdiChild, Plot)
        myPlot.chartPCA.SaveImage(ImageFormat.Emf)
    End If

    If Me.ActiveMdiChild.Name = "chartDendrogram" Then
        Dim myDendrogram As chartDendrogram = CType(Me.ActiveMdiChild,
chartDendrogram)
        myDendrogram.chDendrogram.SaveImage(ImageFormat.Emf)
    End If

    If Me.ActiveMdiChild.Name = "barChart" Then
        Dim mybarChart As barChart = CType(Me.ActiveMdiChild, barChart)
        mybarChart.chartBar.SaveImage(ImageFormat.Emf)
    End If

    If Me.ActiveMdiChild.Name = "Text_Output" Then
        Dim myTextOutput As Text_Output = CType(Me.ActiveMdiChild, Text_Output)
        Dim Selection As String = myTextOutput.dataReport.SelectedText
        Clipboard.SetDataObject(Selection)
    End If

    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Clipboard.SetDataObject(DataTable.DataTable.Clip)
    End If

Case 11
    'Paste
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim data As IDataObject = Clipboard.GetDataObject()
        If data.GetDataPresent(DataFormats.Text) Then
            ' there is, so paste it
            DataTable.DataTable.Select(DataTable.DataTable.Row, DataTable.
DataTable.Col, DataTable.DataTable.Rows.Count - 1, DataTable.DataTable.Cols.Count - 1,
False)
            DataTable.DataTable.Clip = CType(data.GetData(DataFormats.Text),
String)
            DataTable.DataTable.Select(DataTable.DataTable.Row, DataTable.
DataTable.Col)
        End If
    End If
Case 16
    'Bold
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim CellRange As CellRange = DataTable.DataTable.Selection()

```

```

        Dim cellStyle As CellStyle = DataTable.DataTable.Styles.Focus

        If cellStyle.Font.Bold = True And Cellstyle.Font.Italic = True And ✓
        CellStyle.Font.underline = True Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle. ✓
            Regular Or FontStyle.Italic Or FontStyle.Underline)
            CellRange.StyleNew.Font = cellStyle.Font

            ElseIf cellStyle.Font.Bold = True And Cellstyle.Font.Italic = True And ✓
            CellStyle.Font.underline = False Then
                cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle. ✓
                Regular Or FontStyle.Italic)
                CellRange.StyleNew.Font = cellStyle.Font

                ElseIf cellStyle.Font.Bold = True And Cellstyle.Font.Italic = False ✓
                And CellStyle.Font.underline = True Then
                    cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle. ✓
                    Regular Or FontStyle.Underline)
                    CellRange.StyleNew.Font = cellStyle.Font

                    ElseIf cellStyle.Font.Bold = True And Cellstyle.Font.Italic = False ✓
                    And CellStyle.Font.underline = False Then
                        cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle. ✓
                        Regular)
                        CellRange.StyleNew.Font = cellStyle.Font

                        ElseIf cellStyle.Font.Bold = False And Cellstyle.Font.Italic = True ✓
                        And CellStyle.Font.underline = True Then
                            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.Bold ✓
                            Or FontStyle.Italic Or FontStyle.Underline)
                            CellRange.StyleNew.Font = cellStyle.Font

                            ElseIf cellStyle.Font.Bold = False And Cellstyle.Font.Italic = True ✓
                            And CellStyle.Font.underline = False Then
                                cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.Bold ✓
                                Or FontStyle.Italic)
                                CellRange.StyleNew.Font = cellStyle.Font

                                ElseIf cellStyle.Font.Bold = False And Cellstyle.Font.Italic = False ✓
                                And CellStyle.Font.underline = True Then
                                    cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.Bold ✓
                                    Or FontStyle.Underline)
                                    CellRange.StyleNew.Font = cellStyle.Font

                                    ElseIf cellStyle.Font.Bold = False And Cellstyle.Font.Italic = False ✓
                                    And CellStyle.Font.underline = False Then
                                        cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.Bold ✓
                                        )
                                        CellRange.StyleNew.Font = cellStyle.Font

                                End If
                            End If

                        Case 17
                            'Italics
                            If Me.ActiveMdiChild Is Nothing Then
                                Return
                            End If
                            If Me.ActiveMdiChild.Name = "Data_Table" Then
                                Dim DataTable As Data_Table
                                DataTable = CType(Me.ActiveMdiChild, Data_Table)
                                Dim CellRange As CellRange = DataTable.DataTable.Selection
                                Dim cellStyle As CellStyle = DataTable.DataTable.Styles.Focus
                                If cellStyle.Font.Italic = True And Cellstyle.Font.Bold = True And ✓
                                CellStyle.Font.underline = True Then
                                    cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle. ✓
                                    Regular Or FontStyle.Bold Or FontStyle.Underline)

```

```

        CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Italic = True And Cellstyle.Font.Bold = True And
        CellStyle.Font.underline = False Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Regular Or FontStyle.Bold)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Italic = True And Cellstyle.Font.Bold = False
And CellStyle.Font.underline = True Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Regular Or FontStyle.Underline)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Italic = True And Cellstyle.Font.Bold = False
And CellStyle.Font.underline = False Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Regular)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Italic = False And Cellstyle.Font.Bold = True
And CellStyle.Font.underline = True Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Italic Or FontStyle.Bold Or FontStyle.Underline)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Italic = False And Cellstyle.Font.Bold = True
And CellStyle.Font.underline = False Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Italic Or FontStyle.Bold)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Italic = False And Cellstyle.Font.Bold = False
And CellStyle.Font.underline = True Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Italic Or FontStyle.Underline)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Italic = False And Cellstyle.Font.Bold = False
And CellStyle.Font.underline = False Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Italic)
            CellRange.StyleNew.Font = cellStyle.Font

    End If
End If

Case 18
    'Underline
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim CellRange As CellRange = DataTable.DataTable.Selection
        Dim cellStyle As CellStyle = DataTable.DataTable.Styles.Focus
        If cellStyle.Font.Underline = True And Cellstyle.Font.Italic = True
And CellStyle.Font.Bold = True Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Regular Or FontStyle.Italic Or FontStyle.Bold)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Underline = True And Cellstyle.Font.Italic =
True And CellStyle.Font.Bold = False Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Regular Or FontStyle.Italic)

```

```

        CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Underline = True And Cellstyle.Font.Italic =
False And CellStyle.Font.Bold = True Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Regular Or FontStyle.Bold)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Underline = True And Cellstyle.Font.Italic =
False And CellStyle.Font.Bold = False Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Regular)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Underline = False And Cellstyle.Font.Italic =
True And CellStyle.Font.Bold = True Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Underline Or FontStyle.Italic Or FontStyle.Bold)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Underline = False And Cellstyle.Font.Italic =
True And CellStyle.Font.Bold = False Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Underline Or FontStyle.Italic)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Underline = False And Cellstyle.Font.Italic =
False And CellStyle.Font.Bold = True Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Underline Or FontStyle.Bold)
            CellRange.StyleNew.Font = cellStyle.Font

        ElseIf cellStyle.Font.Underline = False And Cellstyle.Font.Italic =
False And CellStyle.Font.Bold = False Then
            cellStyle.Font = New Font(DataTable.DataTable.Font, FontStyle.
Underline)
            CellRange.StyleNew.Font = cellStyle.Font

    End If

End If

Case 23
    'Left Justified
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim Data_Table As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim CellRange As CellRange = DataTable.DataTable.Selection
        Dim cellStyle As CellStyle = DataTable.DataTable.Styles.Focus
        cellStyle.TextAlign = TextAlignEnum.LeftCenter
        CellRange.StyleNew.TextAlign = cellStyle.TextAlign

    End If

Case 24
    'Center Justified
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim Data_Table As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim CellRange As CellRange = DataTable.DataTable.Selection

```

```

        Dim cellStyle As CellStyle = DataTable.DataTable.Styles.Focus
        cellStyle.TextAlign = TextAlignEnum.CenterCenter
        CellRange.StyleNew.TextAlign = cellStyle.TextAlign

    End If

Case 25
    'Right Justified
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Dim DataTable As Data_Table
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim CellRange As CellRange = DataTable.DataTable.Selection
        Dim cellStyle As CellStyle = DataTable.DataTable.Styles.Focus
        cellStyle.TextAlign = TextAlignEnum.RightCenter
        CellRange.StyleNew.TextAlign = cellStyle.TextAlign

    End If

End Select
End Sub

#End Region

Private Sub mnuFileSave_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFileSave.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If
    Dim SaveAsDlg As New SaveFileDialog
    Dim DataTable As New Data_Table
    If Me.ActiveMdiChild.Name = "Data_Table" Then
        DataTable = CType(Me.ActiveMdiChild, Data_Table)

        If myFileName Is Nothing Then
            With SaveAsDlg
                .FileName = ""
                .Filter = "Comma Separated (*.csv)|*.csv|All files (*.*)|*.*"
                .FilterIndex = 1
                If .ShowDialog() = DialogResult.Cancel Then Return
                DataTable.DataTable.SaveGrid(SaveAsDlg.FileName, FileFormatEnum.
TextComma, True)
            End With
            myFileName = SaveAsDlg.FileName
        End If
        DataTable.DataTable.SaveGrid(myFileName, FileFormatEnum.TextComma, True)
    End If

End Sub

Private Sub doc_NewPageSetup(ByVal sender As Cl.ClPrintDocument.ClPrintDocument, ByVal
e As Cl.ClPrintDocument.NewPageSetupEventArgs) Handles doc.NewPageSetup 'ClDocument1.
NewPageStarted.NewPageSetup
    If Me.doc.CurrentPage = 2 Then
        With Me.doc.PageHeader
            .RenderText.Style.TextAlignHorz = Cl.ClPrintDocument.AlignHorzEnum.Right
            .RenderText.Text = "Header - Page [@@PageNo@@] of [@@PageCount@@]"
            .Height = 1
        End With
    End If
End Sub

Private Sub mnuFilePrint_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuFilePrint.Click

```

```

    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If

    If Me.ActiveMdiChild.Name = "Text_Output" Then
        Dim TextOutput As Text_Output = CType(Me.ActiveMdiChild, Text_Output)
        Dim text As String = TextOutput.InputText.ToString
        Dim s As Cl.ClPrintDocument.ClDocStyle
        Dim doc As New ClPrintDocument
        With Me.doc
            .Style.Font = New Font("Times New Roman", 12, FontStyle.Regular)
            With .PageHeader
                '.RenderText.Style.TextAlignHorz = Cl.ClPrintDocument.AlignHorzEnum. ✓
                '.RenderText.Text = "Header - Page [@@PageNo@@] of [@@PageCount@@]"
                .Height = 0
            End With
            With .PageFooter
                '.RenderText.Style.TextAlignHorz = Cl.ClPrintDocument.AlignHorzEnum. ✓
                '.RenderText.Style.TextAlignVert = Cl.ClPrintDocument.AlignVertEnum. ✓
            End With
            .RenderText.Text = "Footer - Page [@@PageNo@@] of [@@PageCount@@]"
        End With
        .StartDoc()
        .Style.TextColor = Color.Black
        '.Style.TextAlignHorz = Cl.ClPrintDocument.AlignHorzEnum.Justify
        .RenderBlockText(text)
        .Style.TextAlignHorz = Cl.ClPrintDocument.AlignHorzEnum.Left
        .EndDoc()
    End With

    Dim aprev As New Final_Report
    aprev.ClPrintPreview1.Document = Me.doc
    aprev.ShowDialog()
    aprev.Dispose()

End If

If Me.ActiveMdiChild.Name = "Data_Table" Then
    Dim DataTable As Data_Table = CType(Me.ActiveMdiChild, Data_Table)
    'Me.InternalPrintGrid(DataTable.DataTable)

    'Count the number of filled in columns (i.e. how many variables).
    Dim k, l, m As Integer
    Dim ColumnData As CellRange
    Dim AdjacentColumnData As CellRange
    Dim rel As New Regex("[0-9]")
    For k = 3 To CType(DataTable.DataTable.Cols.Count, Integer) - 2
        ColumnData = DataTable.DataTable.GetCellRange(1, k, CType(DataTable. ✓
        DataTable.Rows.Count, Integer) - 1, k)
        'provide a counter to make sure all columns are contiguous
        AdjacentColumnData = DataTable.DataTable.GetCellRange(1, k + 1, CType( ✓
        DataTable.DataTable.Rows.Count, Integer) - 1, k + 1)
        If Not rel.Matches(ColumnData.Clip).Count = 0 Then
            l = l + 1
        End If
        'count if columns are not adjacent (i.e. any empty columns in between)
        If Not rel.Matches(ColumnData.Clip).Count = 0 And Not rel.Matches( ✓
        AdjacentColumnData.Clip).Count = 0 Then
            m = m + 1
        End If
    Next
    'Count last column if it has data in it
    k = k + 1

```

```

    If Not rel.Matches(ColumnData.Clip).Count = 0 Then
        l = l + 1
    End If

    Dim Replicates As Integer = CType(DataTable.Replicates, Integer)
    If Replicates = Nothing Then
        Dim ReplicateCells As CellRange
        ReplicateCells = DataTable.DataTable.GetCellRange(1, 2, CType(DataTable.
DataTable.Rows.Count - 1, Integer), 2)
        Dim maxReplicate As Integer
        maxReplicate = CType(DataTable.DataTable.Aggregate(AggregateEnum.Max,
ReplicateCells, AggregateFlags.None), Integer)
        Replicates = maxReplicate

    End If

    Dim endCell As Integer
    Dim SampleCells As CellRange
    For m = 1 To DataTable.DataTable.Rows.Count - 1 Step Replicates
        If CType(DataTable.DataTable(m, 1), String) = "" Then
            endCell = m
            m = DataTable.DataTable.Rows.Count
        End If
    Next

    Dim rows, columns As Integer
    For rows = 0 To endCell - 1
        DataTable.DataTable.Rows(rows).Visible = True
    Next
    For rows = endCell To DataTable.DataTable.Rows.Count - 1
        DataTable.DataTable.Rows(rows).Visible = False
    Next

    For columns = 3 + 1 To DataTable.DataTable.Cols.Count - 1
        DataTable.DataTable.Cols(columns).Visible = False
    Next

    'PUT IN TO TEST FORMATTING
    Dim doc As New ClPrintDocument
    currentGrid = DataTable.DataTable
    MakeDoc(Me.doc, Nothing)
    'MakeFlexPrintDoc(Me.doc, Nothing)
    Dim aprev As New Final_Report
    AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf MakeDoc)
    aprev.ClPrintPreview1.Document = Me.doc
    aprev.ShowDialog()
    RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf MakeDoc)

    currentGrid = Nothing
    aprev.Dispose()

    For rows = endCell To DataTable.DataTable.Rows.Count - 1
        DataTable.DataTable.Rows(rows).Visible = True
    Next

    For columns = 3 + 1 To DataTable.DataTable.Cols.Count - 1
        DataTable.DataTable.Cols(columns).Visible = True
    Next

End If

If Me.ActiveMdiChild.Name = "Text_Output" Then
    Dim myTextOutput As Text_Output = CType(Me.ActiveMdiChild, Text_Output)

End If

If Me.ActiveMdiChild.Name = "barChart" Then

```

```

        Dim barChart As barChart = CType(Me.ActiveMdiChild, barChart)
        Dim doc As New C1PrintDocument
        Doc2D_bar(doc, New GenerateEventArgs)
        Dim aprev As New Final_Report
        AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_bar)
        aprev.C1PrintPreview1.Document = doc
        aprev.ShowDialog()
        RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_bar)
    D_bar)
        aprev.Dispose()
        'barChart.chartBar.PrintChart(PrintScaleEnum.ScaleToFit)

    End If

    If Me.ActiveMdiChild.Name = "chartDendrogram" Then
        Dim myDendrogram As chartDendrogram = CType(Me.ActiveMdiChild, chartDendrogram)
    )
        Dim doc As New C1PrintDocument
        Doc2D_dendrogram(doc, New GenerateEventArgs)
        Dim aprev As New Final_Report
        AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_dendrogram)
    D_dendrogram)
        aprev.C1PrintPreview1.Document = doc
        aprev.ShowDialog()
        RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_dendrogram)
    D_dendrogram)
        aprev.Dispose()
        'barChart.chartBar.PrintChart(PrintScaleEnum.ScaleToFit)
        'myDendrogram.chDendrogram.PrintChart(PrintScaleEnum.ScaleToFit)
    End If

    If Me.ActiveMdiChild.Name = "Plot" Then
        Dim myPlot As Plot = CType(Me.ActiveMdiChild, Plot)
        Dim doc As New C1PrintDocument
        Doc2D_Plot(doc, New GenerateEventArgs)
        Dim aprev As New Final_Report
        AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_Plot)
    )
        aprev.C1PrintPreview1.Document = doc
        aprev.ShowDialog()
        RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_Plot)
    D_Plot)
        aprev.Dispose()
        'myPlot.chartPCA.PrintChart(PrintScaleEnum.ScaleToFit)
    End If

End Sub

Private Sub mnuDataProcessManova_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles mnuDataProcessManova.Click
    If Me.ActiveMdiChild Is Nothing Then
        MessageBox.Show("You have no open data tables with data to process", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If
    If Me.ActiveMdiChild.Name <> "Data_Table" Then
        MessageBox.Show("You must have a Data Table as the active window to process data from", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If

    Dim DataTable As New Data_Table
    DataTable = CType(Me.ActiveMdiChild, Data_Table)
    'Send DataTable to manova class

```

```

Dim myManovaOutput As New manova(DataTable)
'Send results to Text Output

If myManovaOutput.no_Select = False Then
    Return
End If
Dim myManovaTextOutput As New Text_Output
With myManovaTextOutput
    .MdiParent = Me
    .Text = DataTable.TableName & " Manova Output " & Date.Now
    .Show()
    .InputText = myManovaOutput.rich_Text
    .dataReport.Text = myManovaOutput.rich_Text
End With

End Sub

Private Sub mnuDataProcessPCA_Click(ByVal sender As System.Object, ByVal e As System. ✓
EventArgs) Handles mnuDataProcessPCA.Click

    Dim myPCA_Output As New PCA_output.PCA_output_data
    Dim myManova As New manovaexpandtjb.expandtable
    If Me.ActiveMdiChild Is Nothing Then
        MessageBox.Show("You have no open data tables with data to process", "Error", ✓
        MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If
    If Me.ActiveMdiChild.Name <> "Data_Table" Then
        MessageBox.Show("You must have a Data Table as the active window to process ✓
data from", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If

    Dim DataTable As New Data_Table
    DataTable = CType(Me.ActiveMdiChild, Data_Table)

    'Send Table to pca class
    Dim myPCAOutput As New pca(DataTable)
    'Send output to Text and graphs
    If myPCAOutput.no_Select = False Then
        Return
    End If
    Dim myPCATextOutput As New Text_Output
    myPCATextOutput.Inputnewdata = CType(myPCAOutput.TempData, Array)
    With myPCATextOutput
        .MdiParent = Me
        .Text = DataTable.TableName & " PCA Data Output " & Date.Now
        .Show()
        .InputText = myPCAOutput.RichText
        .dataReport.Text = myPCAOutput.RichText
    End With
    'Send variance data to barChart
    Dim BarChart As New barChart
    BarChart.Input_data = CType(myPCAOutput.NewVariances, Array)
    BarChart.Samples = myPCAOutput.Samples
    BarChart.Variables = myPCAOutput.Variables
    BarChart.SampleNames = CType(myPCAOutput.SelectedSamples, Array)
    With BarChart
        .MdiParent = Me
        .Text = DataTable.TableName & " Variances"
        .Show()
    End With
    'Send 1st two components to plot
    Dim Plot As New Plot
    Plot.Input_data = CType(myPCAOutput.TempData, Array)
    Plot.Samples = myPCAOutput.Samples
    Plot.Variables = myPCAOutput.Variables

```

```

        Plot.SampleNames = CType(myPCAOutput.SelectedSamples, Array)
    With Plot
        .MdiParent = Me
        .Text = DataTable.TableName & " Component Loadings"
        .Show()
    End With

End Sub

Private Sub mnuDataProcessCluster_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles mnuDataProcessCluster.Click
    If Me.ActiveMdiChild Is Nothing Then
        MessageBox.Show("You have no open data tables with data to process", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If
    If Me.ActiveMdiChild.Name <> "Data_Table" Then
        MessageBox.Show("You must have a Data Table as the active window to process data from", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If

    Dim DataTable As New Data_Table
    DataTable = CType(Me.ActiveMdiChild, Data_Table)
    'Run Cluster class
    Dim myCluster As New cluster(DataTable)
    If myCluster.no_Select = False Then
        Return
    End If
    'Send data to txt Output
    Dim myClusterTextOutput As New Text_Output
    myClusterTextOutput.Inputnewdata = CType(myCluster.TempData, Array)
    With myClusterTextOutput
        .Text = DataTable.TableName & " Dendrogram Output " & Date.Now
        .Show()
        .MdiParent = Me
        .InputText = myCluster.RichText
        .dataReport.Text = myCluster.RichText
    End With
    'Send data to dendrogram
    Dim Dendrogram As New chartDendrogram
    Dendrogram.Input_data = CType(myCluster.TempData, Array)
    Dendrogram.Samples = myCluster.Samples
    Dendrogram.Variables = myCluster.Variables
    Dendrogram.SampleNames = CType(myCluster.SelectedSamples, Array)
    Dendrogram.AxisLabels = CType(myCluster.AxisLabel, Array)
    With Dendrogram
        .MdiParent = Me
        .Text = DataTable.TableName & " Dendrogram"
        .Show()
    End With

End Sub

Private Sub mnuFormatCellsColor_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles mnuFormatCellsColor.Click
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If

    If Me.ActiveMdiChild.Name = "Data_Table" Then

```

```

        Dim DataTable As Data_Table
        Dim SelectedColor As Color
        DataTable = CType(Me.ActiveMdiChild, Data_Table)
        Dim Selection As CellRange
        Selection = DataTable.DataTable.Selection
        Dim ColorDlg As New ColorDialog
        If ColorDlg.ShowDialog = DialogResult.OK Then
            SelectedColor = ColorDlg.Color
            Selection.StyleNew.BackColor = SelectedColor
        End If
    End If

End Sub

Sub PrintDocumentPages(ByVal firstPage As Integer, ByVal lastPage As Integer)
    Me.currPage = firstPage
    Me.lastPage = lastPage
    Me.PrintDoc = New PrintDocument

    Try
        PrintDoc.Print()
    Catch ex As Exception
        MessageBox.Show(ex.Message, "Print Error")
    End Try

End Sub

Private Sub PrintDoc_PrintPage(ByVal sender As Object, ByVal e As System.Drawing.
    Printing.PrintPageEventArgs) Handles PrintDoc.PrintPage ✓

End Sub

Private Sub mnuFormatChart_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles mnuFormatChart.Click ✓
    If Me.ActiveMdiChild Is Nothing Then
        Return
    End If

    If Me.ActiveMdiChild.Name = "Data_Table" Then
        Return
    End If

    If Me.ActiveMdiChild.Name = "barChart" Then
        Dim barChart As barChart = CType(Me.ActiveMdiChild, barChart)
        barChart.chartBar.ShowProperties()
    End If

    If Me.ActiveMdiChild.Name = "chartDendrogram" Then
        Dim myDendrogram As chartDendrogram = CType(Me.ActiveMdiChild, chartDendrogram)
        myDendrogram.chDendrogram.ShowProperties()
    End If

    If Me.ActiveMdiChild.Name = "Plot" Then
        Dim myPlot As Plot = CType(Me.ActiveMdiChild, Plot)
        myPlot.chartPCA.ShowProperties()
    End If
End Sub

Private Sub FilePrintPreview_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles FilePrintPreview.Click ✓

    If Me.ActiveMdiChild Is Nothing Then

```

```

    Return
End If

If Me.ActiveMdiChild.Name = "Text_Output" Then
    Dim TextOutput As Text_Output = CType(Me.ActiveMdiChild, Text_Output)
    Dim text As String = TextOutput.InputText.ToString
    Dim s As Cl.C1PrintDocument.C1DocStyle
    Dim doc As New ClPrintDocument
    With Me.doc
        .Style.Font = New Font("Times New Roman", 12, FontStyle.Regular)
        With .PageHeader
            '.RenderText.Style.TextAlignHorz = Cl.C1PrintDocument.AlignHorzEnum. ✓
            Right
                '.RenderText.Text = "Header - Page [@@PageNo@@] of [@@PageCount@@]"
                .Height = 0
            End With
        With .PageFooter
            '.RenderText.Style.TextAlignHorz = Cl.C1PrintDocument.AlignHorzEnum. ✓
            Right
                '.RenderText.Style.TextAlignVert = Cl.C1PrintDocument.AlignVertEnum. ✓
            Bottom
                '.RenderText.Text = "Footer - Page [@@PageNo@@] of [@@PageCount@@]"
            End With
        .StartDoc()
        .Style.TextColor = Color.Black
        '.Style.TextAlignHorz = Cl.C1PrintDocument.AlignHorzEnum.Justify
        .RenderBlockText(text)
        .Style.TextAlignHorz = Cl.C1PrintDocument.AlignHorzEnum.Left
        .EndDoc()
    End With

    Dim aprev As New Final_Report
    aprev.C1PrintPreview1.Document = Me.doc
    aprev.ShowDialog()
    aprev.Dispose()

End If

If Me.ActiveMdiChild.Name = "Data_Table" Then
    Dim DataTable As Data_Table = CType(Me.ActiveMdiChild, Data_Table)
    'Me.InternalPrintGrid(DataTable.DataTable)

    'Count the number of filled in columns (i.e. how many variables).
    Dim k, l, m As Integer
    Dim ColumnData As CellRange
    Dim AdjacentColumnData As CellRange
    Dim rel As New Regex("[0-9]")
    For k = 3 To CType(DataTable.DataTable.Cols.Count, Integer) - 2
        ColumnData = DataTable.DataTable.GetCellRange(1, k, CType(DataTable. ✓
        DataTable.Rows.Count, Integer) - 1, k)
        'provide a counter to make sure all columns are contiguous
        AdjacentColumnData = DataTable.DataTable.GetCellRange(1, k + 1, CType( ✓
        DataTable.DataTable.Rows.Count, Integer) - 1, k + 1)
        If Not rel.Matches(ColumnData.Clip).Count = 0 Then
            l = l + 1
        End If
        'count if columns are not adjacent (i.e. any empty columns in between)
        If Not rel.Matches(ColumnData.Clip).Count = 0 And Not rel.Matches( ✓
        AdjacentColumnData.Clip).Count = 0 Then
            m = m + 1
        End If
    Next
    'Count last column if it has data in it
    k = k + 1
    If Not rel.Matches(ColumnData.Clip).Count = 0 Then
        l = l + 1
    End If

```

```

    Dim Replicates As Integer = CType(DataTable.Replicates, Integer)
    If Replicates = Nothing Then
        Dim ReplicateCells As CellRange
        ReplicateCells = DataTable.DataTable.GetCellRange(1, 2, CType(DataTable.
DataTable.Rows.Count - 1, Integer), 2)
        Dim maxReplicate As Integer
        maxReplicate = CType(DataTable.DataTable.Aggregate(AggregateEnum.Max,
ReplicateCells, AggregateFlags.None), Integer)
        Replicates = maxReplicate

    End If

    Dim endCell As Integer
    Dim SampleCells As CellRange
    For m = 1 To DataTable.DataTable.Rows.Count - 1 Step Replicates
        If CType(DataTable.DataTable(m, 1), String) = "" Then
            endCell = m
            m = DataTable.DataTable.Rows.Count
        End If
    Next

    Dim rows, columns As Integer
    For rows = 0 To endCell - 1
        DataTable.DataTable.Rows(rows).Visible = True
    Next
    For rows = endCell To DataTable.DataTable.Rows.Count - 1
        DataTable.DataTable.Rows(rows).Visible = False
    Next

    For columns = 3 + 1 To DataTable.DataTable.Cols.Count - 1
        DataTable.DataTable.Cols(columns).Visible = False
    Next

    'PUT IN TO TEST FORMATTING
    Dim doc As New ClPrintDocument
    currentGrid = DataTable.DataTable
    MakeDoc(Me.doc, Nothing)
    'MakeFlexPrintDoc(Me.doc, Nothing)
    Dim aprev As New Final_Report
    AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf MakeDoc)
    aprev.ClPrintPreview1.Document = Me.doc
    aprev.ShowDialog()
    RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf MakeDoc)

    currentGrid = Nothing
    aprev.Dispose()

    For rows = endCell To DataTable.DataTable.Rows.Count - 1
        DataTable.DataTable.Rows(rows).Visible = True
    Next

    For columns = 3 + 1 To DataTable.DataTable.Cols.Count - 1
        DataTable.DataTable.Cols(columns).Visible = True
    Next

End If

If Me.ActiveMdiChild.Name = "barChart" Then
    Dim barChart As barChart = CType(Me.ActiveMdiChild, barChart)
    Dim doc As New ClPrintDocument
    Doc2D_bar(doc, New GenerateEventArgs)
    Dim aprev As New Final_Report
    AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_bar)
    aprev.ClPrintPreview1.Document = doc

```

```

        aprev.ShowDialog()
        RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2
D_bar)
        aprev.Dispose()
        'barChart.chartBar.PrintChart(PrintScaleEnum.ScaleToFit)

    End If

    If Me.ActiveMdiChild.Name = "chartDendrogram" Then
        Dim myDendrogram As chartDendrogram = CType(Me.ActiveMdiChild, chartDendrogram
)
        Dim doc As New C1PrintDocument
        Doc2D_dendrogram(doc, New GenerateEventArgs)
        Dim aprev As New Final_Report
        AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2
D_dendrogram)
        aprev.C1PrintPreview1.Document = doc
        aprev.ShowDialog()
        RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2
D_dendrogram)
        aprev.Dispose()
        'barChart.chartBar.PrintChart(PrintScaleEnum.ScaleToFit)
        'myDendrogram.chDendrogram.PrintChart(PrintScaleEnum.ScaleToFit)
    End If

    If Me.ActiveMdiChild.Name = "Plot" Then
        Dim myPlot As Plot = CType(Me.ActiveMdiChild, Plot)
        Dim doc As New C1PrintDocument
        Doc2D_Plot(doc, New GenerateEventArgs)
        Dim aprev As New Final_Report
        AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_Plot
)
        aprev.C1PrintPreview1.Document = doc
        aprev.ShowDialog()
        RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2
D_Plot)
        aprev.Dispose()
        'myPlot.chartPCA.PrintChart(PrintScaleEnum.ScaleToFit)

    End If

End Sub

Private Sub Doc2D_Plot(ByVal doc As C1PrintDocument, ByVal e As GenerateEventArgs)
    Dim C1Chart1Raw As Plot = CType(Me.ActiveMdiChild, Plot)
    Dim C1Chart1 As C1.Win.C1Chart.C1Chart = C1Chart1Raw.chartPCA
    With doc
        .DefaultUnit = UnitTypeEnum.Mm
        .StartDoc()
        '.RenderBlockText("Chart", 50, 50, Nothing)
        Dim ww As Double = (CType(.BodyAreaSize.Width, Double)) * 0.9
        .RenderBlockC1Printable(C1Chart1, (.BodyAreaSize.Width * 0.9))
        .CanChangePageMetrics()
        .RenderBlockGraphicsBegin()
        .EndDoc()
    End With
End Sub

Private Sub Doc2D_dendrogram(ByVal doc As C1PrintDocument, ByVal e As
GenerateEventArgs)
    Dim C1Chart1Raw As chartDendrogram = CType(Me.ActiveMdiChild, chartDendrogram)
    Dim C1Chart1 As C1.Win.C1Chart.C1Chart = C1Chart1Raw.chDendrogram
    With doc
        .DefaultUnit = UnitTypeEnum.Mm
        .StartDoc()

```

```

        '.RenderBlockText("Chart", 50, 50, Nothing)
        Dim ww As Double = (CType(.BodyAreaSize.Width, Double)) * 0.9
        .RenderBlockC1Printable(C1Chart1, (.BodyAreaSize.Width * 0.9))
        .CanChangePageMetrics()
        .RenderBlockGraphicsBegin()
        .EndDoc()
    End With
End Sub

Private Sub Doc2D_bar(ByVal doc As C1PrintDocument, ByVal e As GenerateEventArgs)
    Dim C1Chart1Raw As barChart = CType(Me.ActiveMdiChild, barChart)
    Dim C1Chart1 As C1.Win.C1Chart.C1Chart = C1Chart1Raw.chartBar
    With doc
        .DefaultUnit = UnitTypeEnum.Mm
        .StartDoc()
        '.RenderBlockText("Chart", 50, 50, Nothing)
        Dim ww As Double = (CType(.BodyAreaSize.Width, Double)) * 0.9
        .RenderBlockC1Printable(C1Chart1, (.BodyAreaSize.Width * 0.9))
        .CanChangePageMetrics()
        .RenderBlockGraphicsBegin()
        .EndDoc()
    End With
End Sub

Private Sub HelpProvider1_Disposed(ByVal sender As Object, ByVal e As System.EventArgs)
    ' Handles HelpProvider1.Disposed
End Sub

Private Sub mnuEditUndo_Click(ByVal sender As System.Object, ByVal e As System.
    ' Handles mnuEditUndo.Click
    Dim DataTable As Data_Table = DirectCast(Me.ActiveMdiChild, Data_Table)
End Sub

Private Sub mnuDataStatistics_Click(ByVal sender As System.Object, ByVal e As System.
    ' Handles mnuDataStatistics.Click
    Dim Report As New Report_Document
    Report.Show()
End Sub

End Class

```

Public Class About

Inherits System.Windows.Forms.Form

Windows Form Designer generated code

Private Sub About_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles MyBase.Load

End Sub

Private Sub About_MdiChildActivate(ByVal sender As Object, ByVal e As System. ✓
EventArgs) Handles MyBase.MdiChildActivate

End Sub

Private Sub Label3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles Label3.Click

End Sub

End Class

```
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices
```

```
' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.
```

```
' Review the values of the assembly attributes
```

```
<Assembly: AssemblyTitle("")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("")>
<Assembly: AssemblyCopyright("")>
<Assembly: AssemblyTrademark("")>
<Assembly: CLSCompliant(True)>
```

```
'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("3648409D-0530-443F-8B55-1698FC969708")>
```

```
' Version information for an assembly consists of the following four values:
```

```
,
'     Major Version
'     Minor Version
'     Build Number
'     Revision
,
```

```
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
```

```
<Assembly: AssemblyVersion("1.0.*")>
```

```

Imports Cl.Win.ClChart
Imports System.Math
Imports System.Drawing.Imaging
Imports System.Drawing.Printing
Imports Cl.Win.ClPrintPreview
Imports Cl.ClPrintDocument

Public Class barChart
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents chartBar As Cl.Win.ClChart.ClChart
    Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem6 As System.Windows.Forms.MenuItem
    Friend WithEvents ctxCopy As System.Windows.Forms.MenuItem
    Friend WithEvents ctxSaveAs As System.Windows.Forms.MenuItem
    Friend WithEvents ctxPrint As System.Windows.Forms.MenuItem
    Friend WithEvents ctxExit As System.Windows.Forms.MenuItem
    Friend WithEvents ContextMenuBarChart As System.Windows.Forms.ContextMenu
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Dim resources As System.Resources.ResourceManager = New System.Resources.
ResourceManager(GetType(barChart))
        Me.chartBar = New Cl.Win.ClChart.ClChart
        Me.ContextMenuBarChart = New System.Windows.Forms.ContextMenu
        Me.ctxCopy = New System.Windows.Forms.MenuItem
        Me.ctxSaveAs = New System.Windows.Forms.MenuItem
        Me.MenuItem3 = New System.Windows.Forms.MenuItem
        Me.ctxPrint = New System.Windows.Forms.MenuItem
        Me.MenuItem6 = New System.Windows.Forms.MenuItem
        Me.ctxExit = New System.Windows.Forms.MenuItem
        CType(Me.chartBar, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
    End Sub

    'chartBar
    '

    Me.chartBar.BackColor = System.Drawing.Color.White
    Me.chartBar.DataSource = Nothing
    Me.chartBar.Dock = System.Windows.Forms.DockStyle.Fill
    Me.chartBar.Location = New System.Drawing.Point(0, 0)
    Me.chartBar.Name = "chartBar"

```

```
Me.chartBar.PropBag = "<?xml version='1.0'?><Chart2DPropBag Version=''>  
<StyleCollection><NamedStyle><ParentName>Area</ParentName><StyleData>Border=None,Black,1;</StyleData><Name>PlotAr  
&-ea</Name></NamedStyle><NamedStyle><ParentName>Legend.default</ParentName><StyleData>Border=None,Black,1;</StyleData><Name>Legend</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>Border=None,Black,1;</StyleData><Name>Footer</Name></NamedStyle><NamedStyle><ParentName>Area.default</ParentName><StyleData /><Name>Area</Name></NamedStyle><NamedStyle><ParentName>Control.default</ParentName><StyleData>BackColor=White;</StyleData><Name>Control</Name></NamedStyle><NamedStyle><ParentName>Area.default</ParentName><StyleData>Rotation=Rotate0;Border=None,Transparent,1;AlignHorz=Center;BackColor=Transparent;Opaque=False;AlignVert=Bottom;</StyleData><Name>AxisX</Name></NamedStyle><NamedStyle><ParentName>Area.default</ParentName><StyleData>Rotation=Rotate270;Border=None,Transparent,1;AlignHorz=Near;BackColor=Transparent;Opaque=False;AlignVert=Center;</StyleData><Name>AxisY</Name></NamedStyle><NamedStyle><ParentName>LabelStyleDefault.default</ParentName><StyleData /><Name>LabelStyleDefault</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>Border=None,Black,1;Wrap=False;AlignVert=Top;</StyleData><Name>Legend.default</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>Border=None,Black,1;BackColor=Transparent;</StyleData><Name>LabelStyleDefault.default</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>Border=None,Black,1;Font=Microsoft Sans Serif,8.25pt;</StyleData><Name>Header</Name></NamedStyle><NamedStyle><ParentName /><StyleData>ForeColor=ControlText;Border=None,Black,1;BackColor=Control;</StyleData><Name>Control.default</Name></NamedStyle><NamedStyle><ParentName>Area</ParentName><StyleData>Rotation=Rotate90;Border=None,Transparent,1;AlignHorz=Far;BackColor=Transparent;AlignVert=Center;</StyleData><Name>AxisY2</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>Border=None,Black,1;AlignVert=Top;</StyleData><Name>Area.default</Name></NamedStyle></StyleCollection><Header Compass="North"><Text>Percent Variance Exp</Text></Header><Footer Compass="South"><Text /></Footer><Legend Visible=False"><Text /></Legend><ChartArea /><Axes><Axis UnitMajor="1" UnitMinor="0.5" AutoMajor="True" AutoMinor="True" AutoMax="True" AutoMin="True" Ma" x="5" Min="1" _onTop="0" Compass="South"><GridMajor AutoSpace="True" Color="LightGray" Pattern="Dash" Thickness="1" /><GridMinor AutoSpace="True" Color="LightGray" Pattern="Dash" Thickness="1" /><Text /></Axis><Axis UnitMajor="2" UnitMinor="1" AutoMajor="True" AutoMinor="True" AutoMax="True" AutoMin="True" Max
```

```

    = "26" Min & _
    = "8" _onTop="0" Compass="West"><GridMajor AutoSpace="True" Color="
    "LightGray" Pat & _
    "tern="Dash" Thickness="1" /><GridMinor AutoSpace="True" Color="LightGray"
    Patter" & _
    "n="Dash" Thickness="1" /><Text /></Axis><Axis UnitMajor="0" UnitMinor="0"
    AutoMa" & _
    "jor="True" AutoMinor="True" AutoMax="True" AutoMin="True" Max="0" Min=
    "0" _onTop" & _
    = "0" Compass="East"><GridMajor AutoSpace="True" Color="LightGray" Pattern
    ="Dash" & _
    " Thickness="1" /><GridMinor AutoSpace="True" Color="LightGray" Pattern="
    "Dash" Th" & _
    "ickness="1" /><Text /></Axis></Axes><ChartGroupsCollection><ChartGroup>
    <ShowOutl" & _
    "ine>True</ShowOutline><HiLoData>FillFalling=True, FillTransparent=True, FullWidth="
    & _
    "False, ShowClose=True, ShowOpen=True</HiLoData><ChartType>Bar</ChartType><Name>Gro"
    & _
    "upl</Name><Bar>ClusterOverlap=0, ClusterWidth=80</Bar><DataSerializer Hole="3.402
    " & _
    "8234663852886E+38" DefaultSet="True"><DataSeriesCollection>
    <DataSeriesSerializer" & _
    "><SeriesLabel>Variances</SeriesLabel><DataTypes>Double;Double;Double;Double;Doub"
    & _
    "le</DataTypes><DataFields>;;;</DataFields><SymbolStyle Color="Coral" Shape="
    "Box" & _
    "" /><X /><Y1 /><Y /><LineStyle Color="Blue" Pattern="Solid" Thickness="1"
    /><Tag" & _
    " /><Y2 /><Y3 /></DataSeriesSerializer></DataSeriesCollection></DataSerializer><B"
    & _
    "ubble>EncodingMethod=Diameter, MaximumSize=20, MinimumSize=5</Bubble><Pie>OtherOff"
    & _
    "set=0, Start=0</Pie><Polar>Degrees=True, PiRatioAnnotations=True, Start=0</Polar><S"
    & _
    "tacked>False</Stacked><Radar>Degrees=True, Filled=False, Start=0</Radar><Visible>T"
    & _
    "rue</Visible></ChartGroup><ChartGroup><ShowOutline>True</ShowOutline><HiLoData>F"
    & _
    "illFalling=True, FillTransparent=True, FullWidth=False, ShowClose=True, ShowOpen=Tru"
    & _
    "e</HiLoData><ChartType>XYPlot</ChartType><Name>Group2</Name><Bar>ClusterOverlap="
    & _
    "0, ClusterWidth=50</Bar><DataSerializer Hole="3.4028234663852886E+38" /><Bubble>
    E" & _
    "ncodingMethod=Diameter, MaximumSize=20, MinimumSize=5</Bubble><Pie>OtherOffset=0, S"
    & _
    "tart=0</Pie><Polar>Degrees=True, PiRatioAnnotations=True, Start=0</Polar><Stacked>"
    & _
    "False</Stacked><Radar>Degrees=True, Filled=False, Start=0</Radar><Visible>True</Vi"
    & _
    "sible></ChartGroup></ChartGroupsCollection></Chart2DPropBag>"
    Me.chartBar.Size = New System.Drawing.Size(422, 373)
    Me.chartBar.TabIndex = 0
    '
    ' ContextMenuBarChart
    '
    Me.ContextMenuBarChart.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
    ctxCopy, Me.ctxSaveAs, Me.MenuItem3, Me.ctxPrint, Me.MenuItem6, Me.ctxExit})
    '
    ' ctxCopy
    '
    Me.ctxCopy.Index = 0
    Me.ctxCopy.Text = "&Copy"
    '
    ' ctxSaveAs
    '

```

```

Me.ctxSaveAs.Index = 1
Me.ctxSaveAs.Text = "Save &As"
'
'MenuItem3
'
Me.MenuItem3.Index = 2
Me.MenuItem3.Text = "- "
'
'ctxPrint
'
Me.ctxPrint.Index = 3
Me.ctxPrint.Text = "&Print"
'
'MenuItem6
'
Me.MenuItem6.Index = 4
Me.MenuItem6.Text = "- "
'
'ctxExit
'
Me.ctxExit.Index = 5
Me.ctxExit.Text = "E&xit"
'
'barChart
'
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.ClientSize = New System.Drawing.Size(422, 373)
Me.ContextMenu = Me.ContextMenuBarChart
Me.Controls.Add(Me.chartBar)
Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
Me.Name = "barChart"
Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
Me.Text = "barChart"
CType(Me.chartBar, System.ComponentModel.ISupportInitialize).EndInit()
Me.ResumeLayout(False)

```

End Sub

#End Region

```

Private mySamples As Integer
Private myVariables As Integer
Private myInput_data As Array
Private mySampleNames As Array
Private myDataSeries As Integer

```

```

Public Property Variables() As Integer
    Get
        Return myVariables
    End Get
    Set(ByVal Value As Integer)
        myVariables = Value
    End Set
End Property

```

```

Public Property Samples() As Integer
    Get
        Return mySamples
    End Get
    Set(ByVal Value As Integer)
        mySamples = Value
    End Set
End Property

```

```

Public Property Input_data() As Array
    Get
        Return myInput_data
    End Get

```

```

    End Get
    Set(ByVal Value As Array)
        myInput_data = Value
    End Set
End Property

```

```

Public Property SampleNames() As Array
    Get
        Return mySampleNames
    End Get
    Set(ByVal Value As Array)
        mySampleNames = Value
    End Set
End Property

```

```

Public Property DataSeries() As Integer
    Get
        Return myDataSeries
    End Get
    Set(ByVal Value As Integer)
        myDataSeries = Value
    End Set
End Property

```

```

Private Sub mnuFileClose_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs)
    Me.Close()
End Sub

```

```

Private Sub chartBar_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles chartBar.Load

```

```

    Dim chartData As Cl.Win.C1Chart.ChartDataSeries
    Dim chartDataXY As Cl.Win.C1Chart.ChartData
    Dim AxisCounter As Integer = Input_data.Length
    Dim Counter As Integer = 0

```

```

    Dim xAxisData(AxisCounter - 1) As Double
    Dim yAxisDataPercent(AxisCounter - 1) As Double
    Dim sumVariance As Double

```

```

    For Counter = 0 To AxisCounter - 1
        sumVariance = sumVariance + CType(Input_data.GetValue(Counter), Double)
    Next

```

```

    For Counter = 0 To AxisCounter - 1
        xAxisData(Counter) = Counter + 1
        yAxisDataPercent(Counter) = (100 * CType(Input_data.GetValue(Counter),
Double)) / sumVariance

```

```

    Next

```

```

    'Populate Chart
    chartBar.Style.Border.BorderStyle = Cl.Win.C1Chart.BorderStyleEnum.Solid
    chartBar.Style.Border.Thickness = 1
    chartBar.ChartGroups(0).ChartData.SeriesList(0).X.CopyDataIn(xAxisData)
    chartBar.ChartGroups(0).ChartData.SeriesList(0).Y.CopyDataIn(yAxisDataPercent)
    chartBar.ChartArea.AxisX.Text = ControlChars.Lf + "Principal Component"
    chartBar.ChartArea.AxisY.Alignment = StringAlignment.Center
    chartBar.ChartArea.AxisY.Text = "Percent Variance Explained" + ControlChars.Lf + "

```

```

    chartBar.Header.Style.Font = New Font("Arial", 10, FontStyle.Bold)
    chartBar.ChartArea.AxisX.TickMinor = TickMarksEnum.None

```

```

'Determine the percent explained at each bar
Dim percentExplained(Input_data.GetUpperBound(0)) As Double
Dim totalVariance As Double
Dim tempPercent(Input_data.GetUpperBound(0)) As Double
For Counter = 0 To Input_data.GetUpperBound(0)
    totalVariance = totalVariance + CType(Input_data.GetValue(Counter), Double)
Next
For Counter = 0 To Input_data.GetUpperBound(0)
    tempPercent(Counter) = 100 * (CType(Input_data.GetValue(Counter), Double)) /
totalVariance
    If Counter > 0 Then
        percentExplained(Counter) = percentExplained(Counter - 1) + tempPercent
    (Counter)
    ElseIf Counter = 0 Then
        percentExplained(Counter) = tempPercent(Counter)
    End If
    percentExplained(Counter) = Math.Round(percentExplained(Counter), 1)
Next

'Add data labels

Dim cLabs As ChartLabels = chartBar.ChartLabels
cLabs.DefaultLabelStyle.BackColor = Color.White
cLabs.DefaultLabelStyle.Border.BorderStyle = BorderStyleEnum.Empty
cLabs.DefaultLabelStyle.Border.Thickness = 0

For Counter = 0 To percentExplained.GetUpperBound(0)
    Dim cLab As Cl.Win.ClChart.Label = cLabs.LabelsCollection.AddNewLabel()
    cLab.Text = percentExplained(Counter).ToString
    cLab.AttachMethod = AttachMethodEnum.DataIndex
    cLab.AttachMethodData.GroupIndex = 0
    cLab.AttachMethodData.SeriesIndex = 0
    cLab.AttachMethodData.PointIndex = Counter
    cLab.Connected = True
    cLab.Offset = 30
    cLab.Visible = True
    cLab.Compass = LabelCompassEnum.NorthEast
Next

End Sub

Private Sub ctxCopy_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles ctxCopy.Click
    Dim myChart As barChart = Me
    myChart.chartBar.SaveImage(ImageFormat.Emf)
End Sub

Private Sub ctxSaveAs_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles ctxSaveAs.Click ✓
    Dim lastFilterIndex As Integer = 1
    Dim myChart As barChart = Me
    Dim sfg As New SaveFileDialog

    sfg.Filter = "Metafiles (*.emf)|*.emf|" + "Bmp files (*.bmp)|*.bmp|" + "Gif files ✓
(*.gif)|*.gif|" + "Jpeg files (*.jpg;*.jpeg)|*.jpg;*.jpeg|" + "Png files (*.png)|*.png✓
|" + "All graphic files (*.emf;*.bmp;*.gif;*.jpg;*.jpeg;*.png)|*.emf;*.bmp;*.gif;*.jpg✓
;*.jpeg;*.png"
    sfg.FilterIndex = lastFilterIndex
    sfg.OverwritePrompt = True
    sfg.CheckPathExists = True
    sfg.RestoreDirectory = False
    sfg.ValidateNames = True

```

```

    If sfg.ShowDialog() = DialogResult.OK Then
        Dim fn As String = sfg.FileName
        Dim indext As Integer = fn.LastIndexOf(".")c)
        If indext < 0 Then
            indext = fn.Length + 1
            fn += ".emf"
        Else
            indext += 1
        End If
        Dim ext As String = fn.Substring(indext)
        Dim imgfmt As ImageFormat = Nothing

        Select Case ext
            Case "emf"
                imgfmt = ImageFormat.Emf
                myChart.chartBar.SaveImage(fn, imgfmt)

            Case "bmp"
                imgfmt = ImageFormat.Bmp

            Case "gif"
                imgfmt = ImageFormat.Gif

            Case "jpeg", "jpg"
                imgfmt = ImageFormat.Jpeg

            Case "png"
                imgfmt = ImageFormat.Png

            Case Else
                Return
        End Select

        lastFilterIndex = sfg.FilterIndex

        If Not imgfmt.Equals(ImageFormat.Emf) Then
            Dim img As Image = myChart.chartBar.GetImage()
            img.Save(fn, imgfmt)
            img.Dispose()
        End If
        sfg.Dispose()
    End Sub

    Private Sub ctxExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
        Handles ctxExit.Click
            Me.Close()
        End Sub

    Private Sub chartBar_Click(ByVal sender As Object, ByVal e As System.EventArgs) ✓
        Handles chartBar.Click
            Me.Activate()
        End Sub

    Private Sub ctxPrint_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
        Handles ctxPrint.Click
            Dim doc As New C1PrintDocument
            Doc2D_bar(doc, New GenerateEventArgs)
            Dim aprev As New Final_Report
            AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_bar)
            aprev.C1PrintPreview1.Document = doc
            aprev.ShowDialog()
            RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_bar)
            aprev.Dispose()
            'barChart.chartBar.PrintChart(PrintScaleEnum.ScaleToFit)
        End Sub

```

```
Private Sub Doc2D_bar(ByVal doc As ClPrintDocument, ByVal e As GenerateEventArgs)
    Dim ClChart1Raw As barChart = Me
    Dim ClChart1 As Cl.Win.ClChart.ClChart = ClChart1Raw.chartBar
    With doc
        .DefaultUnit = UnitTypeEnum.Mm
        .StartDoc()
        '.RenderBlockText("Chart", 50, 50, Nothing)
        Dim ww As Double = (CType(.BodyAreaSize.Width, Double)) * 0.9
        .RenderBlockClPrintable(ClChart1, (.BodyAreaSize.Width * 0.9))
        .CanChangePageMetrics()
        .RenderBlockGraphicsBegin()
        .EndDoc()
    End With
End Sub
```

```
End Class
```

```
Imports Cl.Win.ClChart
Imports System.Drawing.Imaging
Imports System.Drawing.Printing
Imports Cl.ClPrintDocument
```

```
Public Class chartDendrogram
    Inherits System.Windows.Forms.Form
```

```
#Region " Windows Form Designer generated code "
```

```
    Public Sub New()
        MyBase.New()
```

```
        'This call is required by the Windows Form Designer.
        InitializeComponent()
```

```
        'Add any initialization after the InitializeComponent() call
```

```
    End Sub
```

```
    'Form overrides dispose to clean up the component list.
```

```
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
```

```
        If disposing Then
```

```
            If Not (components Is Nothing) Then
                components.Dispose()
```

```
            End If
```

```
        End If
```

```
        MyBase.Dispose(disposing)
```

```
    End Sub
```

```
    'Required by the Windows Form Designer
```

```
    Private components As System.ComponentModel.IContainer
```

```
    'NOTE: The following procedure is required by the Windows Form Designer
```

```
    'It can be modified using the Windows Form Designer.
```

```
    'Do not modify it using the code editor.
```

```
    Friend WithEvents chDendrogram As Cl.Win.ClChart.ClChart
```

```
    Friend WithEvents ctxCopy As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents ctxSaveAs As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents ctxPrint As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem6 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents ctxExit As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents ContextMenuDendrogram As System.Windows.Forms.ContextMenu
```

```
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
```

```
    Dim resources As System.Resources.ResourceManager = New System.Resources.
    ResourceManager(GetType(chartDendrogram))
```

```
    Me.chDendrogram = New Cl.Win.ClChart.ClChart
```

```
    Me.ContextMenuDendrogram = New System.Windows.Forms.ContextMenu
```

```
    Me.ctxCopy = New System.Windows.Forms.MenuItem
```

```
    Me.ctxSaveAs = New System.Windows.Forms.MenuItem
```

```
    Me.MenuItem3 = New System.Windows.Forms.MenuItem
```

```
    Me.ctxPrint = New System.Windows.Forms.MenuItem
```

```
    Me.MenuItem6 = New System.Windows.Forms.MenuItem
```

```
    Me.ctxExit = New System.Windows.Forms.MenuItem
```

```
    CType(Me.chDendrogram, System.ComponentModel.ISupportInitialize).BeginInit()
```

```
    Me.SuspendLayout()
```

```
    '
```

```
    'chDendrogram
```

```
    '
```

```
    Me.chDendrogram.BackColor = System.Drawing.Color.White
```

```
    Me.chDendrogram.DataSource = Nothing
```

```
    Me.chDendrogram.Dock = System.Windows.Forms.DockStyle.Fill
```

```
    Me.chDendrogram.Location = New System.Drawing.Point(0, 0)
```

```
    Me.chDendrogram.Name = "chDendrogram"
```

```
    Me.chDendrogram.PropBag = "<?xml version='1.0'?'><Chart2DPropBag Version='''''>
<StyleCollection><NamedStyle><Par" &
```

```

entName>Area</ParentName><StyleData>Border=None,Black,1;</StyleData><Name>PlotAr"
&
"ea</Name></NamedStyle><NamedStyle><ParentName>Legend.default</ParentName><StyleD"
&
"ata /><Name>Legend</Name></NamedStyle><NamedStyle><ParentName>Control</ParentNam"
&
"e><StyleData>Border=None,Black,1;</StyleData><Name>Footer</Name></NamedStyle><Na"
&
"medStyle><ParentName>Area.default</ParentName><StyleData /><Name>Area</Name></Na"
&
"medStyle><NamedStyle><ParentName>Control.default</ParentName><StyleData>BackColo"
&
"r=White;</StyleData><Name>Control</Name></NamedStyle><NamedStyle><ParentName>Are"
&
"a</ParentName><StyleData>Rotation=Rotate0;Border=None,Transparent,1;AlignHorz=Ce"
&
"nter;BackColor=Transparent;Opaque=False;AlignVert=Bottom;</StyleData><Name>AxisX"
&
"</Name></NamedStyle><NamedStyle><ParentName>Area</ParentName><StyleData>Rotation"
&
"=Rotate270;Border=None,Transparent,1;AlignHorz=Near;BackColor=Transparent;Opaque"
&
"=False;AlignVert=Center;</StyleData><Name>AxisY</Name></NamedStyle><NamedStyle><"
&
"ParentName>LabelStyleDefault.default</ParentName><StyleData /><Name>LabelStyleDe"
&
"fault</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>"
&
"Border=None,Black,1;Wrap=False;AlignVert=Top;</StyleData><Name>Legend.default</N"
&
"ame></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>Border=N"
&
"one,Black,1;BackColor=Transparent;</StyleData><Name>LabelStyleDefault.default</N"
&
"ame></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>Border=N"
&
"one,Black,1;BackColor=Transparent;</StyleData><Name>Header</Name></NamedStyle><N"
&
"amedStyle><ParentName /><StyleData>ForeColor=ControlText;Border=None,Black,1;Bac"
&
"kColor=Control;</StyleData><Name>Control.default</Name></NamedStyle><NamedStyle>"
&
"<ParentName>Area</ParentName><StyleData>Rotation=Rotate90;Border=None,Transparen"
&
"t,1;AlignHorz=Far;BackColor=Transparent;AlignVert=Center;</StyleData><Name>AxisY"
&
"2</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><StyleData>Bord"
&
"er=None,Black,1;AlignVert=Top;</StyleData><Name>Area.default</Name></NamedStyle>"
&
"</StyleCollection><Header Compass=""North""><Text>Sample Relatedness</Text></
Heade" &
"r><Footer Compass=""South""><Text /></Footer><Legend Visible=""False"" Compass="
"East" &
""><Text /></Legend><ChartArea /><Axes><Axis UnitMajor=""1"" UnitMinor=""0.5""
AutoMa" &
"jor=""True"" AutoMinor=""True"" AutoMax=""True"" AutoMin=""True"" Max=""5"" Min=
"1"" _onTop" &
""0"" Compass=""South""><GridMajor AutoSpace=""True"" Color=""LightGray""
Pattern=""Dash" &
"" Thickness=""1"" /><GridMinor AutoSpace=""True"" Color=""LightGray"" Pattern="
"Dash" T" &
"thickness=""1"" /><Text /></Axis><Axis UnitMajor=""2"" UnitMinor=""1"" AutoMajor=
"True"" &
"AutoMinor=""True"" AutoMax=""True"" AutoMin=""True"" Max=""26"" Min=""8"" _onTop
=""0"" Comp" &
"ass=""West""><GridMajor AutoSpace=""True"" Color=""LightGray"" Pattern=""Dash"

```

```

Thickness" &
"s="1" /><GridMinor AutoSpace="True" Color="LightGray" Pattern="Dash"
Thickness="" &
"1" /><Text /></Axis><Axis UnitMajor="0" UnitMinor="0" AutoMajor="True"
AutoMinor" &
="True" AutoMax="True" AutoMin="True" Max="0" Min="0" _onTop="0"
Compass="East">" &
"<GridMajor AutoSpace="True" Color="LightGray" Pattern="Dash" Thickness="1
" /><Gr" &
"idMinor AutoSpace="True" Color="LightGray" Pattern="Dash" Thickness="1"
/><Text " &
"/></Axis></Axes><ChartGroupsCollection><ChartGroup><ShowOutline>True</ShowOutlin"
&
"e"><HiLoData>FillFalling=True, FillTransparent=True, FullWidth=False, ShowClose=True"
&
", ShowOpen=True</HiLoData><ChartType>XYPlot</ChartType><Name>Group1</Name><Bar>Cl"
&
"usterOverlap=0, ClusterWidth=50</Bar><DataSerializer Hole="3.4028234663852886E+38"
" &
"" DefaultSet="True"><DataSeriesCollection><DataSeriesSerializer><SeriesLabel>
Den" &
"drogram</SeriesLabel><DataTypes>Double;Double;Double;Double;Double</DataTypes><D"
&
"ataFields>;,;,;</DataFields><SymbolStyle Color="Cyan" Shape="None" /><X /><Y1
/><" &
"Y /><LineStyle Color="DarkMagenta" Pattern="Solid" Thickness="1" /><Tag />
<Y2 />" &
"<Y3 /></DataSeriesSerializer></DataSeriesCollection></DataSerializer><Bubble>Enc"
&
"odingMethod=Diameter, MaximumSize=20, MinimumSize=5</Bubble><Pie>OtherOffset=0, Sta"
&
"rt=0</Pie><Polar>Degrees=True, PiRatioAnnotations=True, Start=0</Polar><Stacked>Fa"
&
"lse</Stacked><Radar>Degrees=True, Filled=False, Start=0</Radar><Visible>True</Visi"
&
"ble></ChartGroup><ChartGroup><ShowOutline>True</ShowOutline><HiLoData>FillFallin"
&
"ng=True, FillTransparent=True, FullWidth=False, ShowClose=True, ShowOpen=True</HiLoDa"
&
"ta><ChartType>XYPlot</ChartType><Name>Group2</Name><Bar>ClusterOverlap=0, Cluster"
&
"Width=50</Bar><DataSerializer Hole="3.4028234663852886E+38" /><Bubble>
EncodingMe" &
"thod=Diameter, MaximumSize=20, MinimumSize=5</Bubble><Pie>OtherOffset=0, Start=0</P"
&
"ie><Polar>Degrees=True, PiRatioAnnotations=True, Start=0</Polar><Stacked>False</St"
&
"acked><Radar>Degrees=True, Filled=False, Start=0</Radar><Visible>True</Visible></C"
&
"hartGroup></ChartGroupsCollection></Chart2DPropBag>"
Me.chDendrogram.Size = New System.Drawing.Size(422, 393)
Me.chDendrogram.TabIndex = 0
'
'ContextMenuDendrogram
'
Me.ContextMenuDendrogram.MenuItems.AddRange(New System.Windows.Forms.MenuItem()
{Me.ctxCopy, Me.ctxSaveAs, Me.MenuItem3, Me.ctxPrint, Me.MenuItem6, Me.ctxExit})
'
'ctxCopy
'
Me.ctxCopy.Index = 0
Me.ctxCopy.Text = "&Copy"
'
'ctxSaveAs
'
Me.ctxSaveAs.Index = 1
Me.ctxSaveAs.Text = "Save &As"

```

```

    'MenuItem3
    '
    Me.MenuItem3.Index = 2
    Me.MenuItem3.Text = "-"
    '
    'ctxPrint
    '
    Me.ctxPrint.Index = 3
    Me.ctxPrint.Text = "&Print"
    '
    'MenuItem6
    '
    Me.MenuItem6.Index = 4
    Me.MenuItem6.Text = "-"
    '
    'ctxExit
    '
    Me.ctxExit.Index = 5
    Me.ctxExit.Text = "E&xit"
    '
    'chartDendrogram
    '
    Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
    Me.ClientSize = New System.Drawing.Size(422, 393)
    Me.ContextMenu = Me.ContextMenuDendrogram
    Me.Controls.Add(Me.chDendrogram)
    Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
    Me.Name = "chartDendrogram"
    Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
    Me.Text = "chartDendrogram"
    CType(Me.chDendrogram, System.ComponentModel.ISupportInitialize).EndInit()
    Me.ResumeLayout(False)

```

End Sub

#End Region

```

Private mySamples As Integer
Private myVariables As Integer
Private myInput_data As Array
Private mySampleNames As Array
Private myAxisLabels As Array

```

```

Public Property Variables() As Integer
    Get
        Return myVariables
    End Get
    Set(ByVal Value As Integer)
        myVariables = Value
    End Set
End Property

```

```

Public Property Samples() As Integer
    Get
        Return mySamples
    End Get
    Set(ByVal Value As Integer)
        mySamples = Value
    End Set
End Property

```

```

Public Property Input_data() As Array
    Get
        Return myInput_data
    End Get
    Set(ByVal Value As Array)

```

```

        myInput_data = Value
    End Set
End Property

Public Property SampleNames() As Array
    Get
        Return mySampleNames
    End Get
    Set(ByVal Value As Array)
        mySampleNames = Value
    End Set
End Property

Public Property AxisLabels() As Array
    Get
        Return myAxisLabels
    End Get
    Set(ByVal Value As Array)
        myAxisLabels = Value
    End Set
End Property

Private Sub mnuFileClose_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs)
    Me.Close()
End Sub

Private Sub ClChart1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles chDendrogram.Load

    'List sample numbers
    Dim SampleNumbers As Integer = Me.SampleNames.GetUpperBound(0)
    Dim Counter As Integer
    Dim newArray As Array
    'For Counter = 0 To SampleNumbers - 1
    'Next

    Dim chartData As ClWin.ClChart.ChartDataSeries
    Dim chartDataXY As ClWin.ClChart.ChartData
    Dim chartLabels As ClWin.ClChart.ChartLabels
    Dim chartLabel As Label
    Dim AxisCounter As Integer
    Dim xAxisData(Samples - 1) As Double
    Dim yAxisData(Samples - 1) As Double
    chDendrogram.Style.Border.BorderStyle = ClWin.ClChart.BorderStyleEnum.Solid
    chDendrogram.Style.Border.Thickness = 1
    chDendrogram.ChartArea.AxisX.Text = ControlChars.Lf + "Sample"
    chDendrogram.ChartArea.AxisY.Text = "Distance of Relatedness" + ControlChars.Lf +
"
    chDendrogram.Header.Style.Font = New Font("Arial", 10, FontStyle.Bold)
    chDendrogram.ChartArea.AxisX.TickMinor = TickMarksEnum.None

    'Create new Array with SampleData

    Dim tempAxisLabels(AxisLabels.GetUpperBound(1) - 1) As Integer
    Dim finalAxisLabels(AxisLabels.GetUpperBound(1) - 1) As String
    For Counter = 0 To AxisLabels.GetUpperBound(1) - 1
        tempAxisLabels(Counter) = CType(AxisLabels.GetValue(1, Counter + 1), Integer)
    Next
    Dim tempIndex As Integer
    For Counter = 0 To AxisLabels.GetUpperBound(1) - 1
        tempIndex = CType(tempAxisLabels.GetValue(Counter), Integer)
        finalAxisLabels(Counter) = CType(SampleNames.GetValue(tempIndex - 1), String)
    Next

```

```

'Add labels
With chDendrogram.ChartArea.AxisX
    .AnnoMethod = C1.Win.C1Chart.AnnotationMethodEnum.ValueLabels
    .ValueLabels.Clear()
    .ValueLabels.AddNewLabel()
    .AnnotationRotation = -30
    For Counter = 0 To AxisLabels.GetUpperBound(1) - 1
        .ValueLabels.AddNewLabel()
        .ValueLabels(Counter).Text = finalAxisLabels(Counter)
        .ValueLabels(Counter).NumericValue = Counter + 1
    Next
End With

'Make an Array of only the vertical distances
Dim interDistance(Input_data.GetUpperBound(0)) As Single
For Counter = 0 To CType(Input_data.GetUpperBound(0), Integer)
    interDistance(Counter) = CType(Input_data.GetValue(Counter, 2), Single)
Next

'Create chart area
Dim area As Area = chDendrogram.ChartArea
area.Style.Border.BorderStyle = C1.Win.C1Chart.BorderStyleEnum.None
area.Style.BackColor = Color.Transparent
area.Visible = True

'Create chart group
Dim group As ChartGroup = chDendrogram.ChartGroups(0)
group.ChartType = Chart2DTypeEnum.XYPlot

'Create data and data series
Dim data As ChartData = group.ChartData
Dim s As New ChartDataSeries
data.SeriesList.Add(s)
Dim ps() As PointF

'Copy in a zero point
s = New ChartDataSeries
data.SeriesList.Add(s)
ps = New PointF() {New PointF(0.0F, 0.0F), New PointF(0.0F, 0.0F)}
s.PointData.CopyDataIn(ps)
s.SymbolStyle.Shape = C1.Win.C1Chart.SymbolShapeEnum.None
s.LineStyle.Color = Color.Black

'Place the same two lines on first two Samples
s = New ChartDataSeries
data.SeriesList.Add(s)
ps = New PointF() {New PointF(1.0F, 0.0F), New PointF(1.0F, interDistance(0))}
s.PointData.CopyDataIn(ps)
s.SymbolStyle.Shape = C1.Win.C1Chart.SymbolShapeEnum.None
s.LineStyle.Color = Color.Black
s.LineStyle.Thickness = 2

s = New ChartDataSeries
data.SeriesList.Add(s)
ps = New PointF() {New PointF(2.0F, 0.0F), New PointF(2.0F, interDistance(0))}
s.PointData.CopyDataIn(ps)
s.SymbolStyle.Shape = C1.Win.C1Chart.SymbolShapeEnum.None
s.LineStyle.Color = Color.Black
s.LineStyle.Thickness = 2

'Connect these two vertical lines with a "y-only" line
s = New ChartDataSeries
data.SeriesList.Add(s)
ps = New PointF() {New PointF(1.0F, interDistance(0)), New PointF(2.0F,
interDistance(0))}
s.PointData.CopyDataIn(ps)

```

```

s.SymbolStyle.Shape = C1.Win.C1Chart.SymbolShapeEnum.None
s.LineStyle.Color = Color.Black
s.LineStyle.Thickness = 2

'Populate the rest of the grid with lines and crosses
For Counter = 3 To SampleNames.GetUpperBound(0) + 1
    s = New ChartDataSeries
    data.SeriesList.Add(s)
    ps = New PointF() {New PointF(Counter, 0.0F), New PointF(Counter,
interDistance(Counter - 2))}
    s.PointData.CopyDataIn(ps)
    s.SymbolStyle.Shape = C1.Win.C1Chart.SymbolShapeEnum.None
    s.LineStyle.Color = Color.Black
    s.LineStyle.Thickness = 2

    s = New ChartDataSeries
    data.SeriesList.Add(s)
    ps = New PointF() {New PointF(Counter - 1.5F, interDistance(Counter - 2)), New
PointF(Counter, interDistance(Counter - 2))}
    s.PointData.CopyDataIn(ps)
    s.SymbolStyle.Shape = C1.Win.C1Chart.SymbolShapeEnum.None
    s.LineStyle.Color = Color.Black
    s.LineStyle.Thickness = 2

    s = New ChartDataSeries
    data.SeriesList.Add(s)
    ps = New PointF() {New PointF(Counter - 1.5F, interDistance(Counter - 2)), New
PointF(Counter - 1.5F, interDistance(Counter - 3))}
    s.PointData.CopyDataIn(ps)
    s.SymbolStyle.Shape = C1.Win.C1Chart.SymbolShapeEnum.None
    s.LineStyle.Color = Color.Black
    s.LineStyle.Thickness = 2

Next

'Copy in a zero point at the end
s = New ChartDataSeries
data.SeriesList.Add(s)
ps = New PointF() {New PointF(SampleNames.GetUpperBound(0) + 2, 0.0F), New PointF
(SampleNames.GetUpperBound(0) + 2, 0.0F)}
s.PointData.CopyDataIn(ps)
s.SymbolStyle.Shape = C1.Win.C1Chart.SymbolShapeEnum.None
s.LineStyle.Color = Color.Black

End Sub

Private Sub ctxCopy_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ctxCopy.Click
    Dim myDendrogram As chartDendrogram = Me
    myDendrogram.chDendrogram.SaveImage(ImageFormat.Emf)
End Sub

Private Sub ctxSaveAs_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles ctxSaveAs.Click
    Dim lastFilterIndex As Integer = 1
    Dim myDendrogram As chartDendrogram = Me
    Dim sfg As New SaveFileDialog

    sfg.Filter = "Metafiles (*.emf)|*.emf|" + "Bmp files (*.bmp)|*.bmp|" + "Gif files
(*.gif)|*.gif|" + "Jpeg files (*.jpg;*.jpeg)|*.jpg;*.jpeg|" + "Png files (*.png)|*.png"
| + "All graphic files (*.emf;*.bmp;*.gif;*.jpg;*.jpeg;*.png)|*.emf;*.bmp;*.gif;*.jpg;
*.jpeg;*.png"
    sfg.FilterIndex = lastFilterIndex
    sfg.OverwritePrompt = True
    sfg.CheckPathExists = True
    sfg.RestoreDirectory = False
    sfg.ValidateNames = True

```

```

If sfg.ShowDialog() = DialogResult.OK Then
    Dim fn As String = sfg.FileName
    Dim indext As Integer = fn.LastIndexOf(".")c)
    If indext < 0 Then
        indext = fn.Length + 1
        fn += ".emf"
    Else
        indext += 1
    End If
    Dim ext As String = fn.Substring(indext)
    Dim imgfmt As ImageFormat = Nothing

    Select Case ext
        Case "emf"
            imgfmt = ImageFormat.Emf
            myDendrogram.chDendrogram.SaveImage(fn, imgfmt)

        Case "bmp"
            imgfmt = ImageFormat.Bmp

        Case "gif"
            imgfmt = ImageFormat.Gif

        Case "jpeg", "jpg"
            imgfmt = ImageFormat.Jpeg

        Case "png"
            imgfmt = ImageFormat.Png

        Case Else
            Return
    End Select

    lastFilterIndex = sfg.FilterIndex

    If Not imgfmt.Equals(ImageFormat.Emf) Then
        Dim img As Image = myDendrogram.chDendrogram.GetImage()
        img.Save(fn, imgfmt)
        img.Dispose()
    End If
    End If
    sfg.Dispose()
End Sub

Private Sub ctxExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles ctxExit.Click
    Me.Close()
End Sub

Private Sub chDendrogram_Click(ByVal sender As Object, ByVal e As System.EventArgs) ✓
Handles chDendrogram.Click
    Me.Activate()
End Sub

Private Sub ctxPrint_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles ctxPrint.Click
    Dim doc As New ClPrintDocument
    Doc2D_dendrogram(doc, New GenerateEventArgs)
    Dim aprev As New Final_Report
    AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf
Doc2D_dendrogram) ✓
    aprev.ClPrintPreview1.Document = doc
    aprev.ShowDialog()
    RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf
Doc2D_dendrogram) ✓

```

```
    aprev.Dispose()
    'barChart.chartBar.PrintChart(PrintScaleEnum.ScaleToFit)

End Sub

Private Sub Doc2D_dendrogram(ByVal doc As ClPrintDocument, ByVal e As
GenerateEventArgs)
    Dim ClChart1Raw As chartDendrogram = Me
    Dim ClChart1 As Cl.Win.ClChart.ClChart = ClChart1Raw.chDendrogram
    With doc
        .DefaultUnit = UnitTypeEnum.Mm
        .StartDoc()
        '.RenderBlockText("Chart", 50, 50, Nothing)
        Dim ww As Double = (CType(.BodyAreaSize.Width, Double)) * 0.9
        .RenderBlockClPrintable(ClChart1, (.BodyAreaSize.Width * 0.9))
        .CanChangePageMetrics()
        .RenderBlockGraphicsBegin()
        .EndDoc()
    End With
End Sub
End Class
```

```
Imports System.Text.RegularExpressions
Imports Cl.Win.C1FlexGrid
```

```
Public Class cluster
```

```
Private myLinkages As linkages.pdist_linkage
Private myClusterLinks As clusters_tjb.clusterlinks
Private myDendrogram As dendrogram_tjb.dendrogram_tjb_output
Private myAxisLabel As Object
Private mySamples As Integer
Private myVariables As Integer
Private mySelectedSamples() As Object
Private myTempData As Object
Private myRichText As String
```

```
Public ReadOnly Property AxisLabel() As Object
    Get
```

```
        Return myAxisLabel
```

```
    End Get
```

```
End Property
```

```
Public ReadOnly Property Samples() As Integer
```

```
    Get
```

```
        Return mySamples
```

```
    End Get
```

```
End Property
```

```
Public ReadOnly Property Variables() As Integer
```

```
    Get
```

```
        Return myVariables
```

```
    End Get
```

```
End Property
```

```
Public ReadOnly Property SelectedSamples() As Object
```

```
    Get
```

```
        Return mySelectedSamples
```

```
    End Get
```

```
End Property
```

```
Public ReadOnly Property TempData() As Object
```

```
    Get
```

```
        Return myTempData
```

```
    End Get
```

```
End Property
```

```
Public ReadOnly Property RichText() As String
```

```
    Get
```

```
        Return myRichText
```

```
    End Get
```

```
End Property
```

```
Friend Sub New(ByRef DataTable As Data_Table)
```

```
    Dim myLinkages As New linkages.pdist_linkage
```

```
    Dim myClusterLinks As New clusters_tjb.clusterlinks
```

```
    Dim myDendrogram As New dendrogram_tjb.dendrogram_tjb_output
```

```
    Dim SelectSamples As New Select_Samples
```

```
    Dim i As Integer
```

```
    i = DataTable.Rows.Count - 1
```

```
    'Define a grid with all of the data in column 1 and 2
```

```
    Dim SampleNameList As New Cl.Win.C1FlexGrid.CellRange
```

```

SampleNameList = DataTable.DataTable.GetCellRange(1, 1, i, 1)
Dim newlineString As String = Nothing
Dim Counter As Integer
Dim sampleNameClip As String = SampleNameList.Clip

'Make the clip devoid of whitespace cells
For Counter = 1 To i
    If CType(DataTable.DataTable(Counter, 2), String) = "1" Then
        newlineString = newlineString & CType(DataTable.DataTable(Counter, 1),
String) & Environment.NewLine
    End If
Next

'Open up the Sample Selection Dialog
SelectSamples.samples = newlineString

Try
    SelectSamples.Text = "Choose Samples for Cluster Analysis"
    SelectSamples.ShowDialog()

Catch ex As Exception
    MessageBox.Show(ex.Message, "Error creating dialog", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
End Try

If SelectSamples.DialogResult = DialogResult.Cancel Then
    Return
End If

'Determine which samples where selected and save the names in a String Clip
Dim SelectedSamples As String
SelectedSamples = SelectSamples.SampleChoice()
Dim q As Integer = 0
'Find where the alpha next to \n characters are
Dim re As New Regex("[a-zA-Z0-9]\x0D")
Dim mc As MatchCollection = re.Matches(SelectedSamples)
'Find out how many alpha or numbers next to \n characters there are
q = mc.Count

'Make an array of sample names displayed to user
Dim SelectedSampleArray(q, 1) As String
Dim SampleCounter As Integer
For SampleCounter = 0 To q
    SelectedSampleArray(SampleCounter, 0) = CType(SelectSamples.SelectSamples
(SampleCounter + 1, 1), String)
    SelectedSampleArray(SampleCounter, 1) = CType(SelectSamples.SelectSamples
(SampleCounter + 1, 2), String)
Next

'Make an array of selected samples to be processed
Dim SelectedSamplesUser(q) As String
Dim arraynumber As Integer = 0
For SampleCounter = 0 To q
    If SelectedSampleArray(SampleCounter, 0) = "True" Then
        SelectedSamplesUser(arraynumber) = SelectedSampleArray(SampleCounter, 1)
        arraynumber = arraynumber + 1
    End If
Next

'Check to make sure at least 2 samples were chosen
If arraynumber < 2 Then
    MessageBox.Show("You must select at least two samples", "Sample Selection",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    Return
End If

```

```

'Redimension the selected sample array
ReDim Preserve SelectedSamplesUser(arraynumber - 1)

'Find out how many columns have data in them
Dim NumberofVariables As CellRange
NumberofVariables = DataTable.DataTable.GetCellRange(1, 3, CType(DataTable.
DataTable.Rows.Count, Integer) - 1, CType(DataTable.DataTable.Cols.Count, Integer) -
1)
1) Dim ColumnData, AdjacentColumnData As CellRange
Dim l As Integer = 0
Dim k, m As Integer

'Count the number of filled in columns (i.e. how many variables).
Dim re1 As New Regex("[0-9]")
For k = 3 To CType(DataTable.DataTable.Cols.Count, Integer) - 2
    ColumnData = DataTable.DataTable.GetCellRange(1, k, CType(DataTable.DataTable.
Rows.Count, Integer) - 1, k)
    'provide a counter to make sure all columns are contiguous
    AdjacentColumnData = DataTable.DataTable.GetCellRange(1, k + 1, CType
(DataTable.DataTable.Rows.Count, Integer) - 1, k + 1)
    If Not re1.Matches(ColumnData.Clip).Count = 0 Then
        l = l + 1
    End If
    'count if columns are not adjacent (i.e. any empty columns in between)
    If Not re1.Matches(ColumnData.Clip).Count = 0 And Not re1.Matches
(AdjacentColumnData.Clip).Count = 0 Then
        m = m + 1
    End If
Next
'Count last column if it has data in it
k = k + 1
If Not re1.Matches(ColumnData.Clip).Count = 0 Then
    l = l + 1
End If
'Make user reformat data so the routine will not break
If Not m = l - 1 Then
    MessageBox.Show("It appears that you have a column with missing data. Please
delete or fill in any columns with no data that are inbetween data-bearing columns",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    Return
End If

'Determine the number of replicates
Dim Replicates As Integer = CType(DataTable.Replicates, Integer)
If Replicates = Nothing Then
    Dim ReplicateCells As CellRange
    ReplicateCells = DataTable.DataTable.GetCellRange(1, 2, CType(DataTable.
DataTable.Rows.Count - 1, Integer), 2)
    Dim maxReplicate As Integer
    maxReplicate = CType(DataTable.DataTable.Aggregate(AggregateEnum.Max,
ReplicateCells, AggregateFlags.None), Integer)
    Replicates = maxReplicate

End If

'Create an array to "hold" the averages and STDs of each group of data

Dim x, z As Integer
Dim AverageRange As CellRange
Dim n As Integer = SelectedSamplesUser.GetLength(0)
Dim SamplesToBeProcessed(n - 1) As String
Dim ClusterToBe(n - 1, 1 - 1) As Double
Dim ClusterStdToBe(n - 1, 1 - 1) As Double
For m = 0 To n - 1
    SamplesToBeProcessed(m) = SelectedSamplesUser(m).ToString

```

```

    For k = 1 To CType(DataTable.DataTable.Rows.Count, Integer) - 1
        ColumnData = DataTable.DataTable.GetCellRange(k, 1, k, 1)
        If ColumnData.Clip = SelectedSamplesUser(m) Then
            For z = 3 To 2 + 1
                AverageRange = DataTable.DataTable.GetCellRange(k, z, k +
Replicates - 1, z)
                ClusterToBe(m, z - 3) = CType(DataTable.DataTable.Aggregate
(AggregateEnum.Average, AverageRange, AggregateFlags.None), Double)
                ClusterStdToBe(m, z - 3) = CType(DataTable.DataTable.Aggregate
(AggregateEnum.Std, AverageRange, AggregateFlags.None), Double)
            Next
        End If
    Next
Next

Dim pdist As Object
Dim linkage_output As Object

Try
    Call myLinkages.toms_p_dist(1, pdist, ClusterToBe)
Catch ex As Exception
    'Will catch any error that we're not explicitly trapping.
    MessageBox.Show("Your Data Table has some problem with the 'pdist' routine.
Error message: " & ex.Message, "Serious Data Formatting Problem", MessageBoxButtons.OK
, MessageBoxIcon.Stop)
End Try

Try
    Call myClusterLinks.toms_linkage(1, linkage_output, pdist)
Catch ex As Exception
    'Will catch any error that we're not explicitly trapping.
    MessageBox.Show("Your Data Table has some problem with the 'linkage' routine.
Error message: " & ex.Message, "Serious Data Formatting Problem", MessageBoxButtons.
OK, MessageBoxIcon.Stop)
Return
End Try

Dim axis_label As Object

Try
    Call myDendrogram.dendrogram_output(1, axis_label, linkage_output)
Catch ex As Exception
    'Will catch any error that we're not explicitly trapping.
    MessageBox.Show("Your Data Table has some problem with the 'dendrogram'
routine. Error message: " & ex.Message, "Serious Data Formatting Problem",
MessageBoxButtons.OK, MessageBoxIcon.Stop)
Return
End Try

Dim tempLinkage As Array = CType(linkage_output, Array)
Dim newtempdata(n - 2, 2) As Object
For m = 0 To n - 2
    For k = 0 To 2
        newtempdata(m, k) = tempLinkage.GetValue(m + 1, k + 1)
    Next
Next

'Make a string of axis labels
Dim axis_labelsText As String = ""
Dim axisLabelsTemp As Integer
Dim axis_labels As Array = CType(axis_label, Array)
For Counter = 0 To SelectedSamplesUser.GetUpperBound(0) - 1
    axisLabelsTemp = CType(axis_labels.GetValue(1, Counter + 1), Integer)
    axis_labelsText = axis_labelsText + CType(SelectedSamplesUser.GetValue
(axisLabelsTemp - 1), String) + ControlChars.Lf
Next

```

```

'Make a string of interdistances
Dim interDistanceText As String = ""
Dim interDistanceRound As Single
Dim interDistanceTemp As Double
Dim interDistance(newtempdata.GetUpperBound(0)) As Single
For Counter = 0 To CType(newtempdata.GetUpperBound(0), Integer)
    interDistanceTemp = CType(newtempdata.GetValue(Counter, 2), Double)
    interDistanceRound = CType(Math.Round(interDistanceTemp, 3), Single)
    interDistanceText = interDistanceText + CType(interDistanceRound, String) +
ControlChars.Lf
Next

```

```

Dim richText As String = "The Cluster analysis completed successfully." _
+ ControlChars.Lf + ControlChars.Lf + _
    "In addition to PCA analysis, clustering analysis can be used to determine a
relative 'distance' between relations in multivariate data. This would be analogous to
plotting a family tree and using one inch to represent each generation of distance
between progenitors and progeny. The length of vertical lines in clusters is
indicative of the 'distance' of relatedness between wells. " _
+ ControlChars.Lf + ControlChars.Lf + _
    "The samples, in order of relatedness are listed below: " _
+ ControlChars.Lf + ControlChars.Lf + _
axis_labelsText _
+ ControlChars.Lf + ControlChars.Lf + _
    "The first two are most related, with each after more distantly related. The
distance of relation are given in the Dendrogram plot and below:" _
+ ControlChars.Lf + ControlChars.Lf + _
interDistanceText

```

```

Me.myAxisLabel = axis_label
Me.mySamples = n
Me.myVariables = 1
Me.mySelectedSamples = SelectedSamplesUser
Me.myTempData = newtempdata
Me.myRichText = richText

```

End Sub

End Class

```

Imports Cl.Win.C1FlexGrid
Imports System.Text.RegularExpressions

Public Class Data_Table
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents DataTable As Cl.Win.C1FlexGrid.C1FlexGrid
    Friend WithEvents ContextMenu1 As System.Windows.Forms.ContextMenu
    Friend WithEvents MenuItem5 As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem7 As System.Windows.Forms.MenuItem
    Friend WithEvents mnuContextCut As System.Windows.Forms.MenuItem
    Friend WithEvents mnuContextCopy As System.Windows.Forms.MenuItem
    Friend WithEvents mnuContentPaste As System.Windows.Forms.MenuItem
    Friend WithEvents mnuContextClearContents As System.Windows.Forms.MenuItem
    Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Dim resources As System.Resources.ResourceManager = New System.Resources.
ResourceManager(GetType(Data_Table))
        Me.DataTable = New Cl.Win.C1FlexGrid.C1FlexGrid
        Me.ContextMenu1 = New System.Windows.Forms.ContextMenu
        Me.mnuContextCut = New System.Windows.Forms.MenuItem
        Me.mnuContextCopy = New System.Windows.Forms.MenuItem
        Me.mnuContentPaste = New System.Windows.Forms.MenuItem
        Me.MenuItem5 = New System.Windows.Forms.MenuItem
        Me.mnuContextClearContents = New System.Windows.Forms.MenuItem
        Me.MenuItem7 = New System.Windows.Forms.MenuItem
        Me.MenuItem1 = New System.Windows.Forms.MenuItem
        CType(Me.DataTable, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
    End Sub

    'DataTable
    Me.DataTable.AccessibleDescription = ""
    Me.DataTable.AccessibleName = "Data_Table"
    Me.DataTable.AllowAddNew = True
    Me.DataTable.AllowDelete = True
    Me.DataTable.AllowDragging = Cl.Win.C1FlexGrid.AllowDraggingEnum.None
    Me.DataTable.AllowResizing = Cl.Win.C1FlexGrid.AllowResizingEnum.Both
    Me.DataTable.AllowSorting = Cl.Win.C1FlexGrid.AllowSortingEnum.None

```

85

```

    "or:ControlDarkDark;ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) &
    "Subtotal5{BackColor:ControlDarkDark;ForeCol" &
    "or:White;}" & Microsoft.VisualBasic.ChrW(9))
    Me.DataTable.SubtotalPosition = Cl.Win.ClFlexGrid.SubtotalPositionEnum.BelowData
    Me.DataTable.TabIndex = 0
    '
    'ContextMenu1
    '
    Me.ContextMenu1.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
mnuContextCut, Me.mnuContextCopy, Me.mnuContentPaste, Me.MenuItem1, Me.MenuItem5, Me.
mnuContextClearContents, Me.MenuItem7})
    '
    'mnuContextCut
    '
    Me.mnuContextCut.Index = 0
    Me.mnuContextCut.Text = "Cu&t"
    '
    'mnuContextCopy
    '
    Me.mnuContextCopy.Index = 1
    Me.mnuContextCopy.Text = "&Copy"
    '
    'mnuContentPaste
    '
    Me.mnuContentPaste.Index = 2
    Me.mnuContentPaste.Text = "&Paste"
    '
    'MenuItem5
    '
    Me.MenuItem5.Index = 4
    Me.MenuItem5.Text = "&Delete Column(s)"
    '
    'mnuContextClearContents
    '
    Me.mnuContextClearContents.Index = 5
    Me.mnuContextClearContents.Text = "Clear Co&ntents"
    '
    'MenuItem7
    '
    Me.MenuItem7.Index = 6
    Me.MenuItem7.Text = "- "
    '
    'MenuItem1
    '
    Me.MenuItem1.Index = 3
    Me.MenuItem1.Text = "&Insert Column(s)"
    '
    'Data_Table
    '
    Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
    Me.ClientSize = New System.Drawing.Size(715, 429)
    Me.ContextMenu = Me.ContextMenu1
    Me.Controls.Add(Me.DataTable)
    Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
    Me.Name = "Data_Table"
    Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
    Me.Text = "Data Table"
    CType(Me.DataTable, System.ComponentModel.ISupportInitialize).EndInit()
    Me.ResumeLayout(False)

End Sub

#End Region

Private mTableName As String
Public Property TableName() As String
    Get

```

```

        Return mTableName
    End Get
    Set(ByVal Value As String)
        mTableName = Value
    End Set
End Property

Private mReplicates As String

Public Property Replicates() As String
    Get
        Return CType(mReplicates, String)
    End Get
    Set(ByVal Value As String)
        mReplicates = Value
    End Set
End Property

```

```

Private mColumnHeaders As String
Public Property ColumnHeaders() As String
    Get
        Return CType(mColumnHeaders, String)
    End Get
    Set(ByVal Value As String)
        mColumnHeaders = Value
    End Set
End Property

```

#Region "Data Table initial setup"

```

Private Sub Data_Table_Load(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MyBase.Load

```

```

    'Determine the number of active Data Tables
    Dim NumberOfForms As Array
    NumberOfForms = MdiParent.MdiChildren
    'Increment the number of the active form based on ones already open
    Dim i As Integer
    i = NumberOfForms.GetUpperBound(0)
    Me.Text = "Data Table " & i + 1
    TableName = Me.Text

```

```

    'Place sample name and replicate in column headers if no variable names are
specified

```

```

    If Me.ColumnHeaders = "" Then
        Me.ColumnHeaders = "|SampleID|Replicate"
        Dim cols As String = Me.ColumnHeaders
        'Setup the column split character as |
        Dim colNames As String() = cols.Split(CType("|", Char))
        Dim z As Integer
        'Fill from the third column with the names
        For z = 1 To 2
            DataTable(0, z) = colNames(z)
            DataTable.Cols(z).Name = colNames(z)
        Next
    End If

```

```

    'If user chooses to name columns with variable names
    If Not Me.ColumnHeaders = "" Then
        'set up columns
        'Find out how many individual variable names exist
        Me.ColumnHeaders = "|SampleID|Replicate|" & Me.ColumnHeaders
        Dim q As Integer = 0
        'Find where the \n characters are
        Dim re As New Regex("\x0D")
        Dim mc As MatchCollection = re.Matches(Me.ColumnHeaders)
    End If

```

```

'Find out how many \n characters there are
q = mc.Count
'Replace the \n characters with |
Me.ColumnHeaders = re.Replace(Me.ColumnHeaders, "\x0D", "|")

' set up columns
Dim cols As String = Me.ColumnHeaders
'Setup the column split character as |
Dim colNames As String() = cols.Split(CType("|", Char))
Dim z As Integer
'Fill from the third column with the names
For z = 1 To q
    DataTable(0, z) = colNames(z)
    DataTable.Cols(z).Name = colNames(z)
Next
End If

'The following formatting applies to all rows and columns

'Populate the 0 column rows with row number
Dim y As Integer
Dim rowNames As String
For y = 1 To CType(DataTable.Rows.Count, Integer) - 1
    DataTable.Rows(y).Caption = CType(y, String)
Next
'Populate the replicates column with user specified number of replicates.
Dim countColumn As Integer
For countColumn = 1 To CType(Me.Replicates, Integer)
    For y = countColumn To CType(DataTable.Rows.Count, Integer) - 1
        DataTable.SetData(y, 2, CType(countColumn, String))
        y = y + (CType(Me.Replicates, Integer) - 1)
    Next
Next
'Format each first replicate number to left justify
Dim cs As CellStyle = DataTable.Styles.Add("First")
cs.TextAlign = TextAlignEnum.LeftCenter
Dim CountCell As Integer
For CountCell = 0 To CType(DataTable.Rows.Count, Integer) - 1
    If Val(DataTable(CountCell, 2)) = 1 Then
        DataTable.SetCellStyle(CountCell, 2, cs)
    End If
Next

'Set the replicate number column to be non-editable
DataTable.Cols(2).AllowEditing = False

'Set column data type to Double for each data input column
For countColumn = 3 To CType(DataTable.Cols.Count, Integer) - 1
    DataTable.Cols(countColumn).DataType = GetType(Double)
Next

Dim temp As Object = DataTable.GetType.GetProperties()

```

End Sub

#End Region

```

Private Sub Data Table_ValidateEdit(ByVal sender As Object, ByVal e As
ValidateEditEventArgs) Handles DataTable.ValidateEdit
    ' validate amounts to make sure they are del 13 C values
    If DataTable.Cols(e.Col).DataType Is GetType(Double) Then
        Try
            Dim dbl As Double = Double.Parse(DataTable.Editor.Text())
            If dbl < -100 Or dbl > 60 Then

```

```

        MessageBox.Show("Value does not appear to be a PDB stardardized
isotope value, please try again", "Error")
        e.Cancel = True
    End If
Catch
    e.Cancel = True
End Try
End If
End Sub

```

#Region "Hot keys (copy, cut, paste, delete) events "

```

Private Sub DataTable_KeyDown(ByVal sender As Object, ByVal e As KeyEventArgs) Handles
DataTable.KeyDown
    Dim copy As Boolean, paste As Boolean, cut As Boolean
    ' ** copy: ctrl-C, ctrl-X, ctrl-ins
    If e.Control Then
        If e.KeyCode = Keys.C Or _
            e.KeyCode = Keys.Insert Then
            copy = True
        End If
        If e.KeyCode = Keys.X Then
            cut = True
        End If
    End If
    ' ** paste: ctrl-V, shift-ins
    If (e.Control = True And e.KeyCode = Keys.V) Or _
        (e.Shift And e.KeyCode = Keys.Insert) Then
        paste = True
    End If
    ' ** copy selection to clipboard
    If copy Then
        Clipboard.SetDataObject(DataTable.Clip)
    End If
    ' ** cut selection to the clipboard
    If cut Then
        Clipboard.SetDataObject(DataTable.Clip)
        Dim selected As Cl.Win.ClFlexGrid.CellRange
        selected = DataTable.Selection
        selected.Data = Nothing
    End If
    ' ** paste from clipboard
    If paste Then
        ' see of there's text in the clipboard
        Dim data As IDataObject = Clipboard.GetDataObject()
        If data.GetDataPresent(DataFormats.Text) Then
            ' there is, so paste it
            DataTable.Select(DataTable.Row, DataTable.Col, DataTable.Rows.Count - 1,
DataTable.Cols.Count - 1, False)
            DataTable.Clip = CType(data.GetData(DataFormats.Text), String)
            DataTable.Select(DataTable.Row, DataTable.Col)
        End If
    End If

    'If the user presses the delete key in a cell or in a range of cells, delete them
    If e.KeyCode = Keys.Delete Then
        Dim selected As Cl.Win.ClFlexGrid.CellRange
        selected = DataTable.Selection
        selected.Data = Nothing
    End If
End Sub

```

#End Region

```

Private Sub mnuContextCut_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles mnuContextCut.Click
    Clipboard.SetDataObject(DataTable.Clip)

```

```

        Dim selected As Cl.Win.ClFlexGrid.CellRange
        selected = DataTable.Selection
        selected.Data = Nothing
    End Sub

    Private Sub mnuContextCopy_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles mnuContextCopy.Click
        Clipboard.SetDataObject(DataTable.Clip)
    End Sub

    Private Sub mnuContentPaste_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles mnuContentPaste.Click
        Dim data As IDataObject = Clipboard.GetDataObject()
        If data.GetDataPresent(DataFormats.Text) Then
            ' there is, so paste it
            DataTable.Select(DataTable.Row, DataTable.Col, DataTable.Rows.Count - 1,
            DataTable.Cols.Count - 1, False)
            DataTable.Clip = CType(data.GetData(DataFormats.Text), String)
            DataTable.Select(DataTable.Row, DataTable.Col)
        End If
    End Sub

    Private Sub mnuContextClearContents_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles mnuContextClearContents.Click
        Dim selected As Cl.Win.ClFlexGrid.CellRange
        selected = DataTable.Selection
        selected.Data = Nothing
    End Sub

    Private Sub Data_Table_CellChanged(ByVal sender As Object, ByVal e As RowColEventArgs)
    Handles DataTable.CellChanged
        Dim CellRange As CellRange = Me.DataTable.Selection()
        Dim cellStyle As CellStyle = Me.DataTable.Styles.Focus

        cellStyle.Font = New Font(Me.DataTable.Font, FontStyle.Regular)
        CellRange.StyleNew.Font = cellStyle.Font
    End Sub

    Private Sub MenuItem1_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MenuItem1.Click
        Dim DataTable As Data_Table = Me
        Dim selectedColumns As CellRange
        selectedColumns = DataTable.DataTable.Selection
        Dim selectedColumnLower As Integer = selectedColumns.c1
        Dim selectedColumnUpper As Integer = selectedColumns.c2
        Dim columnRange As ColumnCollection
        columnRange = DataTable.DataTable.Cols
        columnRange.DefaultSize = 70
        Dim columnCount As Integer
        For columnCount = selectedColumnLower To selectedColumnUpper
            columnRange.Insert(columnCount)
        Next
    End Sub

    Private Sub MenuItem5_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MenuItem5.Click
        Dim DataTable As Data_Table = Me
        Dim selectedColumns As CellRange
        selectedColumns = DataTable.DataTable.Selection
        Dim selectedColumnLower As Integer = selectedColumns.c1
        Dim selectedColumnUpper As Integer = selectedColumns.c2
        Dim columnRange As ColumnCollection
        columnRange = DataTable.DataTable.Cols

```

```
        columnRange.DefaultSize = 70
        Dim columnCount As Integer
        For columnCount = selectedColumnLower To selectedColumnUpper
            columnRange.Remove(columnCount)
        Next

    End Sub

    Private Sub DataTable_EnterCell(ByVal sender As Object, ByVal e As System.EventArgs) ✓
        Handles DataTable.EnterCell
            Dim CellRange As CellRange = Me.DataTable.Selection()
            Dim cellStyle As CellStyle = Me.DataTable.Styles.Focus

            cellStyle.Font = New Font(Me.DataTable.Font, FontStyle.Regular)
            CellRange.StyleNew.Font = cellStyle.Font
        End Sub
    End Class
```

```

Public Class Final_Report
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents ClPrintPreview1 As Cl.Win.ClPrintPreview.ClPrintPreview
    Friend WithEvents PreviewToolBarButton1 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton2 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton3 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton4 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton5 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton6 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton7 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton8 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton9 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton10 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton11 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton12 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton13 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton14 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton15 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton16 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton17 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton18 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton19 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton20 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton21 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton22 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton23 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton24 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton25 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton26 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton27 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton28 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton29 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton30 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton31 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton32 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    Friend WithEvents PreviewToolBarButton33 As Cl.Win.ClPrintPreview.PreviewToolBarButton
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Dim resources As System.Resources.ResourceManager = New System.Resources.

```

```

ResourceManager(GetType(Final_Report))
    Me.C1PrintPreview1 = New C1.Win.C1PrintPreview.C1PrintPreview
    Me.PreviewToolBarButton1 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton2 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton3 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton4 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton5 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton6 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton7 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton8 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton9 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton10 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton11 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton12 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton13 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton14 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton15 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton16 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton17 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton18 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton19 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton20 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton21 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton22 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton23 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton24 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton25 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton26 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton27 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton28 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton29 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton30 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton31 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton32 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    Me.PreviewToolBarButton33 = New C1.Win.C1PrintPreview.PreviewToolBarButton
    CType(Me.C1PrintPreview1, System.ComponentModel.ISupportInitialize).BeginInit()
    Me.SuspendLayout()
    '
    'C1PrintPreview1
    '
    Me.C1PrintPreview1.C1DPageSettings = "color:False;landscape:False;margins:100,100,100,100;papersize:850,1100,TABLAHQAdA" & _
    "BLAHIA"
    Me.C1PrintPreview1.Dock = System.Windows.Forms.DockStyle.Fill
    Me.C1PrintPreview1.Location = New System.Drawing.Point(0, 0)
    Me.C1PrintPreview1.Name = "C1PrintPreview1"
    Me.C1PrintPreview1.NavigationBar.Cursor = System.Windows.Forms.Cursors.Default
    Me.C1PrintPreview1.NavigationBar.Font = New System.Drawing.Font("Microsoft Sans
Serif", 8.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
    Me.C1PrintPreview1.NavigationBar.OutlineView.Cursor = System.Windows.Forms.Cursors
.Default
    Me.C1PrintPreview1.NavigationBar.OutlineView.Font = New System.Drawing.Font(
"Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular, System.Drawing.
GraphicsUnit.Point, CType(0, Byte))
    Me.C1PrintPreview1.NavigationBar.OutlineView.Indent = 19
    Me.C1PrintPreview1.NavigationBar.OutlineView.ItemHeight = 16
    Me.C1PrintPreview1.NavigationBar.OutlineView.TabIndex = 0
    Me.C1PrintPreview1.NavigationBar.OutlineView.Visible = False
    Me.C1PrintPreview1.NavigationBar.Padding = New System.Drawing.Point(6, 3)
    Me.C1PrintPreview1.NavigationBar.TabIndex = 2
    Me.C1PrintPreview1.NavigationBar.ThumbnailsView.AutoArrange = True
    Me.C1PrintPreview1.NavigationBar.ThumbnailsView.Cursor = System.Windows.Forms.
Cursors.Default
    Me.C1PrintPreview1.NavigationBar.ThumbnailsView.Font = New System.Drawing.Font(
"Microsoft Sans Serif", 8.25!, System.Drawing.FontStyle.Regular, System.Drawing.
GraphicsUnit.Point, CType(0, Byte))

```

```

Me.C1PrintPreview1.NavigationBar.ThumbnailsView.TabIndex = 0
Me.C1PrintPreview1.NavigationBar.ThumbnailsView.Visible = True
Me.C1PrintPreview1.NavigationBar.Width = 160
Me.C1PrintPreview1.PreviewPane.ZoomFactor = 0.75!
Me.C1PrintPreview1.PreviewPane.ZoomMode = C1.Win.C1PrintPreview.ZoomModeEnum. ✓
Custom
Me.C1PrintPreview1.Size = New System.Drawing.Size(752, 733)
Me.C1PrintPreview1.Splitter.Cursor = System.Windows.Forms.Cursors.VSplit
Me.C1PrintPreview1.Splitter.Width = 3
Me.C1PrintPreview1.StatusBar.Cursor = System.Windows.Forms.Cursors.Default
Me.C1PrintPreview1.StatusBar.Font = New System.Drawing.Font("Microsoft Sans Serif" ✓
, 8.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, ✓
Byte))
Me.C1PrintPreview1.StatusBar.TabIndex = 4
Me.C1PrintPreview1.TabIndex = 0
Me.C1PrintPreview1.ToolBar.Buttons.AddRange(New System.Windows.Forms.ToolBarButton ✓
() {Me.PreviewToolBarButton1, Me.PreviewToolBarButton2, Me.PreviewToolBarButton3, Me. ✓
PreviewToolBarButton4, Me.PreviewToolBarButton5, Me.PreviewToolBarButton6, Me. ✓
PreviewToolBarButton7, Me.PreviewToolBarButton8, Me.PreviewToolBarButton9, Me. ✓
PreviewToolBarButton10, Me.PreviewToolBarButton11, Me.PreviewToolBarButton12, Me. ✓
PreviewToolBarButton13, Me.PreviewToolBarButton14, Me.PreviewToolBarButton15, Me. ✓
PreviewToolBarButton16, Me.PreviewToolBarButton17, Me.PreviewToolBarButton18, Me. ✓
PreviewToolBarButton19, Me.PreviewToolBarButton20, Me.PreviewToolBarButton21, Me. ✓
PreviewToolBarButton22, Me.PreviewToolBarButton23, Me.PreviewToolBarButton24, Me. ✓
PreviewToolBarButton25, Me.PreviewToolBarButton26, Me.PreviewToolBarButton27, Me. ✓
PreviewToolBarButton28, Me.PreviewToolBarButton29, Me.PreviewToolBarButton30, Me. ✓
PreviewToolBarButton31, Me.PreviewToolBarButton32, Me.PreviewToolBarButton33})
Me.C1PrintPreview1.ToolBar.Cursor = System.Windows.Forms.Cursors.Default
Me.C1PrintPreview1.ToolBar.Font = New System.Drawing.Font("Microsoft Sans Serif", ✓
8.25!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, ✓
Byte))
'PreviewToolBarButton1
'
Me.PreviewToolBarButton1.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
FileOpen
Me.PreviewToolBarButton1.ImageIndex = 0
Me.PreviewToolBarButton1.ToolTipText = "File Open"
'PreviewToolBarButton2
'
Me.PreviewToolBarButton2.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
FileSave
Me.PreviewToolBarButton2.ImageIndex = 1
Me.PreviewToolBarButton2.ToolTipText = "File Save"
'PreviewToolBarButton3
'
Me.PreviewToolBarButton3.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
FilePrint
Me.PreviewToolBarButton3.ImageIndex = 2
Me.PreviewToolBarButton3.ToolTipText = "Print"
'PreviewToolBarButton4
'
Me.PreviewToolBarButton4.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
PageSetup
Me.PreviewToolBarButton4.ImageIndex = 3
Me.PreviewToolBarButton4.ToolTipText = "Page Setup"
'PreviewToolBarButton5
'
Me.PreviewToolBarButton5.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
Reflow
Me.PreviewToolBarButton5.ImageIndex = 4
Me.PreviewToolBarButton5.ToolTipText = "Reflow"
'

```

```

        'PreviewToolBarButton6
        '
        Me.PreviewToolBarButton6.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
Stop
        Me.PreviewToolBarButton6.ImageIndex = 5
        Me.PreviewToolBarButton6.ToolTipText = "Stop"
        Me.PreviewToolBarButton6.Visible = False
        '
        'PreviewToolBarButton7
        '
        Me.PreviewToolBarButton7.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
None
        Me.PreviewToolBarButton7.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
        '
        'PreviewToolBarButton8
        '
        Me.PreviewToolBarButton8.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
ShowNavigationBar
        Me.PreviewToolBarButton8.ImageIndex = 6
        Me.PreviewToolBarButton8.Pushed = True
        Me.PreviewToolBarButton8.Style = System.Windows.Forms.ToolBarButtonStyle. ✓
ToggleButton
        Me.PreviewToolBarButton8.ToolTipText = "Show Navigation Bar"
        '
        'PreviewToolBarButton9
        '
        Me.PreviewToolBarButton9.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
None
        Me.PreviewToolBarButton9.Style = System.Windows.Forms.ToolBarButtonStyle.Separator
        '
        'PreviewToolBarButton10
        '
        Me.PreviewToolBarButton10.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
MouseHand
        Me.PreviewToolBarButton10.ImageIndex = 7
        Me.PreviewToolBarButton10.Pushed = True
        Me.PreviewToolBarButton10.Style = System.Windows.Forms.ToolBarButtonStyle. ✓
ToggleButton
        Me.PreviewToolBarButton10.ToolTipText = "Hand Tool"
        '
        'PreviewToolBarButton11
        '
        Me.PreviewToolBarButton11.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
MouseZoom
        Me.PreviewToolBarButton11.ImageIndex = 8
        Me.PreviewToolBarButton11.Style = System.Windows.Forms.ToolBarButtonStyle. ✓
DropDownButton
        Me.PreviewToolBarButton11.ToolTipText = "Zoom In Tool"
        '
        'PreviewToolBarButton12
        '
        Me.PreviewToolBarButton12.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
MouseZoomOut
        Me.PreviewToolBarButton12.ImageIndex = 25
        Me.PreviewToolBarButton12.Style = System.Windows.Forms.ToolBarButtonStyle. ✓
DropDownButton
        Me.PreviewToolBarButton12.ToolTipText = "Zoom Out Tool"
        Me.PreviewToolBarButton12.Visible = False
        '
        'PreviewToolBarButton13
        '
        Me.PreviewToolBarButton13.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum. ✓
MouseSelect
        Me.PreviewToolBarButton13.ImageIndex = 9
        Me.PreviewToolBarButton13.Style = System.Windows.Forms.ToolBarButtonStyle. ✓
ToggleButton
        Me.PreviewToolBarButton13.ToolTipText = "Select Text"

```

```

        'PreviewToolBarButton14
        Me.PreviewToolBarButton14.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. FindText
        Me.PreviewToolBarButton14.ImageIndex = 10
        Me.PreviewToolBarButton14.ToolTipText = "Find Text"
        'PreviewToolBarButton15
        Me.PreviewToolBarButton15.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. None
        Me.PreviewToolBarButton15.Style = System.Windows.Forms.ToolBarButtonStyle. Separator
        'PreviewToolBarButton16
        Me.PreviewToolBarButton16.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. GoFirst
        Me.PreviewToolBarButton16.Enabled = False
        Me.PreviewToolBarButton16.ImageIndex = 11
        Me.PreviewToolBarButton16.ToolTipText = "First Page"
        'PreviewToolBarButton17
        Me.PreviewToolBarButton17.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. GoPrev
        Me.PreviewToolBarButton17.Enabled = False
        Me.PreviewToolBarButton17.ImageIndex = 12
        Me.PreviewToolBarButton17.ToolTipText = "Previous Page"
        'PreviewToolBarButton18
        Me.PreviewToolBarButton18.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. GoNext
        Me.PreviewToolBarButton18.ImageIndex = 13
        Me.PreviewToolBarButton18.ToolTipText = "Next Page"
        'PreviewToolBarButton19
        Me.PreviewToolBarButton19.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. GoLast
        Me.PreviewToolBarButton19.ImageIndex = 14
        Me.PreviewToolBarButton19.ToolTipText = "Last Page"
        'PreviewToolBarButton20
        Me.PreviewToolBarButton20.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. None
        Me.PreviewToolBarButton20.Style = System.Windows.Forms.ToolBarButtonStyle. Separator
        'PreviewToolBarButton21
        Me.PreviewToolBarButton21.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. HistoryPrev
        Me.PreviewToolBarButton21.Enabled = False
        Me.PreviewToolBarButton21.ImageIndex = 15
        Me.PreviewToolBarButton21.ToolTipText = "Previous View"
        Me.PreviewToolBarButton21.Visible = False
        'PreviewToolBarButton22
        Me.PreviewToolBarButton22.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. HistoryNext
        Me.PreviewToolBarButton22.Enabled = False
        Me.PreviewToolBarButton22.ImageIndex = 16

```

```

Me.PreviewToolBarButton22.ToolTipText = "Next View"
Me.PreviewToolBarButton22.Visible = False
'PreviewToolBarButton23
Me.PreviewToolBarButton23.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. None
Me.PreviewToolBarButton23.Style = System.Windows.Forms.ToolBarButtonStyle. Separator
Me.PreviewToolBarButton23.Visible = False
'PreviewToolBarButton24
Me.PreviewToolBarButton24.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. ZoomOut
Me.PreviewToolBarButton24.ImageIndex = 17
Me.PreviewToolBarButton24.ToolTipText = "Zoom Out"
Me.PreviewToolBarButton24.Visible = False
'PreviewToolBarButton25
Me.PreviewToolBarButton25.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. ZoomIn
Me.PreviewToolBarButton25.ImageIndex = 18
Me.PreviewToolBarButton25.ToolTipText = "Zoom In"
Me.PreviewToolBarButton25.Visible = False
'PreviewToolBarButton26
Me.PreviewToolBarButton26.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. None
Me.PreviewToolBarButton26.Style = System.Windows.Forms.ToolBarButtonStyle. Separator
Me.PreviewToolBarButton26.Visible = False
'PreviewToolBarButton27
Me.PreviewToolBarButton27.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. ViewActualSize
Me.PreviewToolBarButton27.ImageIndex = 19
Me.PreviewToolBarButton27.Style = System.Windows.Forms.ToolBarButtonStyle. ToggleButton
Me.PreviewToolBarButton27.ToolTipText = "Actual Size"
'PreviewToolBarButton28
Me.PreviewToolBarButton28.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. ViewFullPage
Me.PreviewToolBarButton28.ImageIndex = 20
Me.PreviewToolBarButton28.Style = System.Windows.Forms.ToolBarButtonStyle. ToggleButton
Me.PreviewToolBarButton28.ToolTipText = "Full Page"
'PreviewToolBarButton29
Me.PreviewToolBarButton29.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. ViewPageWidth
Me.PreviewToolBarButton29.ImageIndex = 21
Me.PreviewToolBarButton29.Style = System.Windows.Forms.ToolBarButtonStyle. ToggleButton
Me.PreviewToolBarButton29.ToolTipText = "Page Width"
'PreviewToolBarButton30
Me.PreviewToolBarButton30.Action = Cl.Win.ClPrintPreview.ToolBarButtonActionEnum. ViewTwoPages
Me.PreviewToolBarButton30.ImageIndex = 22

```

C:\Documents and Settings\tjb\My Documents\...Projects\IRMS Processing_OO\Final_Report.vb 7

```
Me.PreviewToolBarButton30.Style = System.Windows.Forms.ToolBarButtonStyle.
ToggleButton
Me.PreviewToolBarButton30.ToolTipText = "Two Pages"
'
'PreviewToolBarButton31
Me.PreviewToolBarButton31.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum.
ViewFourPages
Me.PreviewToolBarButton31.ImageIndex = 23
Me.PreviewToolBarButton31.Style = System.Windows.Forms.ToolBarButtonStyle.
DropDownButton
Me.PreviewToolBarButton31.ToolTipText = "Four Pages"
'
'PreviewToolBarButton32
Me.PreviewToolBarButton32.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum.
None
Me.PreviewToolBarButton32.Style = System.Windows.Forms.ToolBarButtonStyle.
Separator
Me.PreviewToolBarButton32.Visible = False
'
'PreviewToolBarButton33
Me.PreviewToolBarButton33.Action = C1.Win.C1PrintPreview.ToolBarButtonActionEnum.
Help
Me.PreviewToolBarButton33.ImageIndex = 24
Me.PreviewToolBarButton33.ToolTipText = "Help"
Me.PreviewToolBarButton33.Visible = False
'
'Final_Report
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.ClientSize = New System.Drawing.Size(752, 733)
Me.Controls.Add(Me.C1PrintPreview1)
Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
Me.Name = "Final_Report"
Me.Text = "Print Preview"
CType(Me.C1PrintPreview1, System.ComponentModel.ISupportInitialize).EndInit()
Me.ResumeLayout(False)
```

End Sub

#End Region

```
Private Sub C1PrintPreview1_Load(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles C1PrintPreview1.Load
```

End Sub

End Class

```

Public Class Make_Table
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents tabSetupSpreadsheet As System.Windows.Forms.TabPage
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents btnVariableNameNo As System.Windows.Forms.Button
    Friend WithEvents btnVariableNameYes As System.Windows.Forms.Button
    Friend WithEvents tabcntrSetupData As System.Windows.Forms.TabControl
    Friend WithEvents Panel1 As System.Windows.Forms.Panel
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents txtReplicateNumber As System.Windows.Forms.TextBox
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Dim resources As System.Resources.ResourceManager = New System.Resources.
ResourceManager(GetType(Make_Table))
        Me.tabSetupSpreadsheet = New System.Windows.Forms.TabPage
        Me.Panel1 = New System.Windows.Forms.Panel
        Me.Label3 = New System.Windows.Forms.Label
        Me.txtReplicateNumber = New System.Windows.Forms.TextBox
        Me.Label2 = New System.Windows.Forms.Label
        Me.Label1 = New System.Windows.Forms.Label
        Me.btnVariableNameNo = New System.Windows.Forms.Button
        Me.btnVariableNameYes = New System.Windows.Forms.Button
        Me.tabcntrSetupData = New System.Windows.Forms.TabControl
        Me.tabSetupSpreadsheet.SuspendLayout()
        Me.Panel1.SuspendLayout()
        Me.tabcntrSetupData.SuspendLayout()
        Me.SuspendLayout()
        '
        'tabSetupSpreadsheet
        '
        Me.tabSetupSpreadsheet.Controls.Add(Me.Panel1)
        Me.tabSetupSpreadsheet.Controls.Add(Me.Label2)
        Me.tabSetupSpreadsheet.Controls.Add(Me.Label1)
        Me.tabSetupSpreadsheet.Controls.Add(Me.btnVariableNameNo)
        Me.tabSetupSpreadsheet.Controls.Add(Me.btnVariableNameYes)
        Me.tabSetupSpreadsheet.Location = New System.Drawing.Point(4, 22)
        Me.tabSetupSpreadsheet.Name = "tabSetupSpreadsheet"
        Me.tabSetupSpreadsheet.Size = New System.Drawing.Size(344, 267)
    End Sub

```

```

Me.tabSetupSpreadsheet.TabIndex = 0
Me.tabSetupSpreadsheet.Text = "Setup Data Table"
'
'Panel1
'
Me.Panel1.Controls.Add(Me.Label3)
Me.Panel1.Controls.Add(Me.txtReplicateNumber)
Me.Panel1.Location = New System.Drawing.Point(24, 8)
Me.Panel1.Name = "Panel1"
Me.Panel1.Size = New System.Drawing.Size(296, 56)
Me.Panel1.TabIndex = 0
'
'Label3
'
Me.Label3.Location = New System.Drawing.Point(22, 21)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(176, 23)
Me.Label3.TabIndex = 1
Me.Label3.Text = "Analytical Replicates (Default = 3)"
'
'txtReplicateNumber
'
Me.txtReplicateNumber.Location = New System.Drawing.Point(206, 19)
Me.txtReplicateNumber.Name = "txtReplicateNumber"
Me.txtReplicateNumber.Size = New System.Drawing.Size(32, 20)
Me.txtReplicateNumber.TabIndex = 0
Me.txtReplicateNumber.Text = "3"
Me.txtReplicateNumber.TextAlign = System.Windows.Forms.HorizontalAlignment.Center
'
'Label2
'
Me.Label2.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.75!, System.
Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label2.Location = New System.Drawing.Point(48, 122)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(248, 88)
Me.Label2.TabIndex = 3
Me.Label2.Text = "In other words, would you like each compound to be identified in
the data table (" & _
"will not change results of the analysis, but may be more descriptive if the data"
& _
" table is printed). "
Me.Label2.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
'
'Label1
'
Me.Label1.Font = New System.Drawing.Font("Times New Roman", 14.25!, System.Drawing
.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label1.Location = New System.Drawing.Point(48, 72)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(248, 48)
Me.Label1.TabIndex = 2
Me.Label1.Text = "Would you like to name each of your variables?"
Me.Label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
'
'btnVariableNameNo
'
Me.btnVariableNameNo.DialogResult = System.Windows.Forms.DialogResult.Cancel
Me.btnVariableNameNo.Location = New System.Drawing.Point(200, 224)
Me.btnVariableNameNo.Name = "btnVariableNameNo"
Me.btnVariableNameNo.TabIndex = 1
Me.btnVariableNameNo.Text = "&No"
'
'btnVariableNameYes
'
Me.btnVariableNameYes.DialogResult = System.Windows.Forms.DialogResult.OK
Me.btnVariableNameYes.Location = New System.Drawing.Point(56, 224)

```

```

Me.btnVariableNameYes.Name = "btnVariableNameYes"
Me.btnVariableNameYes.TabIndex = 0
Me.btnVariableNameYes.Text = "&Yes"
'
'tabcntrSetupData
'
Me.tabcntrSetupData.Controls.Add(Me.tabSetupSpreadsheet)
Me.tabcntrSetupData.Dock = System.Windows.Forms.DockStyle.Fill
Me.tabcntrSetupData.ItemSize = New System.Drawing.Size(96, 18)
Me.tabcntrSetupData.Location = New System.Drawing.Point(0, 0)
Me.tabcntrSetupData.Name = "tabcntrSetupData"
Me.tabcntrSetupData.SelectedIndex = 0
Me.tabcntrSetupData.ShowToolTips = True
Me.tabcntrSetupData.Size = New System.Drawing.Size(352, 293)
Me.tabcntrSetupData.SizeMode = System.Windows.Forms.TabSizeMode.Fixed
Me.tabcntrSetupData.TabIndex = 0
'
'Make_Table
'
Me.AcceptButton = Me.btnVariableNameYes
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.CancelButton = Me.btnVariableNameNo
Me.ClientSize = New System.Drawing.Size(352, 293)
Me.Controls.Add(Me.tabcntrSetupData)
Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
Me.MaximizeBox = False
Me.MinimizeBox = False
Me.Name = "Make_Table"
Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
Me.Text = "Make Data Table"
Me.tabSetupSpreadsheet.ResumeLayout(False)
Me.Panell1.ResumeLayout(False)
Me.tabcntrSetupData.ResumeLayout(False)
Me.ResumeLayout(False)

```

End Sub

#End Region

```

Private Sub btnVariableNameNo_Click(ByVal sender As System.Object, ByVal e As System. ✓
EventArgs) Handles btnVariableNameNo.Click

```

End Sub

End Class

```
Imports System.Text.RegularExpressions
Imports Cl.Win.ClFlexGrid
```

```
Public Class manova
```

```
Private myManova As manovaexpandtjb.expandtable
Private myManova_p As manova_probability.manova_p
Private myManova_stats As manova_stats.manova_stats
Private myManova_stats_expanded As manova_expand_stats.manova_expand_statistics
Private myManova_statsFunctions As manova_expand_stats.manova_expand_statistics
```

```
Private Within As Object
Private Between As Object
Private Total As Object
Private dfWithin As Object
Private dfBetween As Object
Private dfTotal As Object
Private Lambda As Object
Private Chisq As Object
Private Chisqdf As Object
Private Eigenval As Object
Private Eigenvec As Object
Private Canon As Object
Private Mdist As Object
Private Gnames As Object
Private testsString As String
Private richText As String
```

```
Public ReadOnly Property outWithin() As Object
    Get
        Return Within
    End Get
End Property
```

```
Public ReadOnly Property outBetween() As Object
    Get
        Return Between
    End Get
End Property
```

```
Public ReadOnly Property outTotal() As Object
    Get
        Return Total
    End Get
End Property
```

```
Public ReadOnly Property outLambda() As Object
    Get
        Return Lambda
    End Get
End Property
```

```
Public ReadOnly Property outChisq() As Object
    Get
        Return Chisq
    End Get
End Property
```

```
Public ReadOnly Property outChisqdf() As Object
    Get
        Return Chisqdf
    End Get
End Property
```

```
Public ReadOnly Property outEigenval() As Object
    Get
```

```

        Return Eigenval
    End Get
End Property

Public ReadOnly Property outEigenvec() As Object
    Get
        Return Eigenvec
    End Get
End Property

Public ReadOnly Property outCanon() As Object
    Get
        Return Canon
    End Get
End Property

Public ReadOnly Property outMdist() As Object
    Get
        Return Mdist
    End Get
End Property

Public ReadOnly Property outGnames() As Object
    Get
        Return Gnames
    End Get
End Property

Public ReadOnly Property tests_String() As String
    Get
        Return testsString
    End Get
End Property

Public ReadOnly Property rich_Text() As String
    Get
        Return richText
    End Get
End Property

Friend Sub New(ByRef DataTable As Data_Table)

    myManova_p = New manova_probability.manova_p

    'Find out how many rows there are
    Dim i As Integer
    i = DataTable.DataTable.Rows.Count - 1
    'Define a grid with all of the data in column 1 and 2
    Dim SelectSamples As New Select_Samples
    Dim SampleNameList As New Cl.Win.ClFlexGrid.CellRange
    SampleNameList = DataTable.DataTable.GetCellRange(1, 1, i, 1)
    Dim newlineString As String = Nothing
    Dim Counter As Integer
    Dim sampleNameClip As String = SampleNameList.Clip

    'Make the clip devoid of whitespace cells
    For Counter = 1 To i
        If CType(DataTable.DataTable(Counter, 2), String) = "1" Then
            newlineString = newlineString & CType(DataTable.DataTable(Counter, 1),
String) & Environment.NewLine
        End If
    Next

    'Open up the Sample Selection Dialog
    SelectSamples.samples = newlineString

```

```

SelectSamples.Text = "Choose Samples for MANOVA"

Try
    SelectSamples.ShowDialog()

Catch ex As Exception
    MessageBox.Show(ex.Message, "Error creating dialog", MessageBoxButtons.OK,
    MessageBoxIcon.Exclamation)

End Try
If SelectSamples.DialogResult = DialogResult.Cancel Then
    Return
End If

'Determine which samples where selected and save the names in a String Clip
Dim SelectedSamples As String
SelectedSamples = SelectSamples.SampleChoice()
Dim q As Integer = 0
'Find where the alpha next to \n characters are
Dim re As New Regex("[a-zA-Z0-9]\x0D")
Dim mc As MatchCollection = re.Matches(SelectedSamples)
'Find out how many alpha or numbers next to \n characters there are
q = mc.Count

'Make an array of sample names displayed to user
Dim SelectedSampleArray(q, 1) As String
Dim SampleCounter As Integer
For SampleCounter = 0 To q
    SelectedSampleArray(SampleCounter, 0) = CType(SelectSamples.SelectSamples
(SampleCounter + 1, 1), String)
    SelectedSampleArray(SampleCounter, 1) = CType(SelectSamples.SelectSamples
(SampleCounter + 1, 2), String)

Next

'Make an array of selected samples to be processed
Dim SelectedSamplesUser(q) As String
Dim arraynumber As Integer = 0
For SampleCounter = 0 To q
    If SelectedSampleArray(SampleCounter, 0) = "True" Then
        SelectedSamplesUser(arraynumber) = SelectedSampleArray(SampleCounter, 1)
        arraynumber = arraynumber + 1
    End If
Next

'Check to make sure at least 2 samples were chosen
If arraynumber < 2 Then
    MessageBox.Show("You must select at least two samples", "Sample Selection",
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    Return
End If

'Redimension the selected sample array
ReDim Preserve SelectedSamplesUser(arraynumber - 1)

'Find out how many columns have data in them
Dim NumberOfVariables As CellRange
NumberOfVariables = DataTable.DataTable.GetCellRange(1, 3, CType(DataTable.
DataTable.Rows.Count, Integer) - 1, CType(DataTable.DataTable.Cols.Count, Integer) -
1)
Dim ColumnData, AdjacentColumnData As CellRange
Dim l As Integer = 0
Dim k, m As Integer

'Count the number of filled in columns (i.e. how many variables).
Dim rel As New Regex("[0-9]")

```

```

For k = 3 To CType(DataTable.DataTable.Cols.Count, Integer) - 2
    ColumnData = DataTable.DataTable.GetCellRange(1, k, CType(DataTable.DataTable.
Rows.Count, Integer) - 1, k)
    'provide a counter to make sure all columns are contiguous
    AdjacentColumnData = DataTable.DataTable.GetCellRange(1, k + 1, CType
(DataTable.DataTable.Rows.Count, Integer) - 1, k + 1)
    If Not rel.Matches(ColumnData.Clip).Count = 0 Then
        l = l + 1
    End If
    'count if columns are not adjacent (i.e. any empty columns in between)
    If Not rel.Matches(ColumnData.Clip).Count = 0 And Not rel.Matches
(AdjacentColumnData.Clip).Count = 0 Then
        m = m + 1
    End If
Next
'Count last column if it has data in it
k = k + 1
If Not rel.Matches(ColumnData.Clip).Count = 0 Then
    l = l + 1
End If
'Make user reformat data so the routine will not break
If Not m = l - 1 Then
    MessageBox.Show("It appears that you have a column with missing data. Please
delete or fill in any columns with no data that are inbetween data-bearing columns",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    Return
End If

'Determine the number of replicates
Dim Replicates As Integer = CType(DataTable.Replicates, Integer)
If Replicates = Nothing Then
    Dim ReplicateCells As CellRange
    ReplicateCells = DataTable.DataTable.GetCellRange(1, 2, CType(DataTable.
DataTable.Rows.Count - 1, Integer), 2)
    Dim maxReplicate As Integer
    maxReplicate = CType(DataTable.DataTable.Aggregate(AggregateEnum.Max,
ReplicateCells, AggregateFlags.None), Integer)
    Replicates = maxReplicate

End If

'Create an array to "hold" the to-be-processed data

Dim x, z As Integer
Dim n As Integer = SelectedSamplesUser.GetLength(0)
Dim SamplesToBeProcessed(n - 1) As String
Dim ManovaExpandables(n - 1) As Object
Dim ManovaExpandToBe(Replicates - 1, l - 1) As Double
For m = 0 To n - 1
    SamplesToBeProcessed(m) = SelectedSamplesUser(m).ToString
    For k = 1 To CType(DataTable.DataTable.Rows.Count, Integer) - 1
        ColumnData = DataTable.DataTable.GetCellRange(k, 1, k, 1)
        If ColumnData.Clip = SelectedSamplesUser(m) Then
            For x = 0 To Replicates - 1
                For z = 3 To 2 + 1
                    ManovaExpandToBe(x, z - 3) = CType(DataTable.DataTable(k + x,
z), Double)
                Next
            Next
        End If
    Next
    ManovaExpandables(m) = ManovaExpandToBe.Clone
Next
Dim myManovaStats As New manova_stats.manova_stats
Dim myManova As New manovaexpandtjb.expandtable
Dim SelectedSamplesExpanded(n - 1) As Object

```

```

Dim SelectedSamplesExpandedStats(n - 1) As Object
For m = 0 To n - 1
    Try
        Call myManova.manova_numbers_expand_no_stats(1, SelectedSamplesExpanded(m) ✓
, ManovaExpandables(m), 1 - 1)
        Catch exp As Exception
            ' Will catch any error that we're not explicitly trapping.
            MessageBox.Show("Your data table is not setup correctly... Error message:✓
" & exp.Message, "Serious Data Formatting Problem", MessageBoxButtons.OK, ✓
MessageBoxIcon.Stop)
        End Try
    Try
        Call myManovaStats.manova_numbers_expand_stats(1, ✓
SelectedSamplesExpandedStats(m), ManovaExpandables(m), 1 - 1)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Serious Error", MessageBoxButtons.OK)
        End Try
    Next

'Determine the number of expanded rows

If SelectedSamplesExpanded.Length < 2 Then
    MessageBox.Show("You must select at least two samples for this test", "Error", ✓
MessageBoxButtons.OK, MessageBoxIcon.Hand)
    Return
End If
Dim tempArray As Array = CType(SelectedSamplesExpanded(0), Array)
Dim ManovaReplicates As Integer = tempArray.GetLength(0) 'Length of expanded
Dim ManovaArrays As Integer = SelectedSamplesUser.GetLength(0) 'Number of Samples
Dim SelectedSamplesUserExpanded((ManovaArrays * ManovaReplicates) - 1) As Object

'Create an array of expanded sample names
Dim CounterExpanded As Integer
Dim index As Integer = 0
For CounterExpanded = 0 To ManovaArrays - 1
    For Counter = index To (index + ManovaReplicates) - 1
        SelectedSamplesUserExpanded(Counter) = SamplesToBeProcessed ✓
(CounterExpanded)
    Next
    index += 10
Next

'Determine the number of combinations of sample tests
Dim NumberOfTests As Integer = 0
For Counter = 0 To ManovaArrays - 1
    NumberOfTests = NumberOfTests + Counter
Next

'Make new arrays with all the test combinations
Dim tempArray2 As Array
Dim SampleNameTest((ManovaReplicates * 2) - 1) As Object
Dim SampleDataTest(((ManovaReplicates * 2) - 1), 1 - 1) As Double
Dim SampleDataTestArray(NumberOfTests - 1) As Object
Dim SampleNameTestArray(NumberOfTests - 1) As Object
CounterExpanded = 0
Dim TestCounter As Integer = 0
Dim PyramidCounter As Integer = 1
Dim TestArrayCounter As Integer = 0
For TestCounter = 0 To ManovaArrays - 1
    For Counter = PyramidCounter To ManovaArrays - 1
        tempArray = CType(SelectedSamplesExpanded(TestCounter), Array)
        tempArray2 = CType(SelectedSamplesExpanded(Counter), Array)

        'Add tempArray into new array
        Array.Copy(tempArray, 1, SampleDataTest, 0, tempArray.Length)
        'Add tempArray2 into new array after tempArray

```

```

Length)    Array.Copy(tempArray2, 1, SampleDataTest, tempArray.Length, tempArray2.  ✓

    'Add sample names to new Array

    For index = 0 To ManovaReplicates - 1
        SampleNameTest(index) = SelectedSamplesUser(TestCounter)
    Next

    For index = ManovaReplicates To SampleNameTest.Length - 1
        SampleNameTest(index) = SelectedSamplesUser(Counter)
    Next

    CounterExpanded += ManovaReplicates * 2

    'Clone these into a the counted Array
    SampleDataTestArray(TestArrayCounter) = SampleDataTest.Clone
    SampleNameTestArray(TestArrayCounter) = SampleNameTest.Clone
    'Now, increment the TestArrayCounter
    TestArrayCounter += 1

Next
    'Increment the PyramidCounter
    PyramidCounter += 1
Next

Dim myManovaExpandedStats As New manova_expand_stats.manova_expand_statistics
Dim myManovaStatistics As New manova_stats_functions.manova_stats_funct
Dim ManovaProb(NumberOfTests - 1) As Object
Dim ManovaStats(NumberOfTests - 1) As Object
Dim ManovaStatsTemp As Object

'Dim all of the Manova stats individually
Dim Within(NumberOfTests - 1) As Object
Dim WithinTemp As Object
Dim Between(NumberOfTests - 1) As Object
Dim BetweenTemp As Object
Dim Total(NumberOfTests - 1) As Object
Dim TotalTemp As Object
Dim dfWithin(NumberOfTests - 1) As Object
Dim dfWithinTemp As Object
Dim dfBetween(NumberOfTests - 1) As Object
Dim dfBetweenTemp As Object
Dim dfTotal(NumberOfTests - 1) As Object
Dim dfTotalTemp As Object
Dim lambda(NumberOfTests - 1) As Object
Dim lambdaTemp As Object
Dim chisq(NumberOfTests - 1) As Object
Dim chisqTemp As Object
Dim chisqdf(NumberOfTests - 1) As Object
Dim chisqdfTemp As Object
Dim eigenval(NumberOfTests - 1) As Object
Dim eigenvalTemp As Object
Dim eigenvec(NumberOfTests - 1) As Object
Dim eigenvecTemp As Object
Dim canon(NumberOfTests - 1) As Object
Dim canonTemp As Object
Dim mdist(NumberOfTests - 1) As Object
Dim mdistTemp As Object
Dim gnames(NumberOfTests - 1) As Object
Dim gnamesTemp As Object

Dim SampleDataTestTempArray As Object
Dim SampleNamesTestTempArray As Object
Dim ManovaProbTemp As Object

```

```

For index = 0 To NumberOfTests - 1
    SampleDataTestTempArray = SampleDataTestArray(index)
    SampleNamesTestTempArray = SampleNameTestArray(index)

    Try
        Call myManova_p.toms_manova_p(1, ManovaProbTemp, SampleNamesTestTempArray, ✓
SampleDataTestTempArray)
        Catch exp As Exception
            'Will catch any error that we're not explicitly trapping.
            MessageBox.Show(exp.message, "Error", MessageBoxButtons.OKCancel, ✓
MessageBoxIcon.Stop)
            Return
        End Try
        ManovaProb(index) = ManovaProbTemp

    Try
        Call myManovaExpandedStats.toms_manova_stats(1, ManovaStatsTemp, ✓
SampleNamesTestTempArray, SampleDataTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon. ✓
Error)
            Return
        End Try
        ManovaStats(index) = ManovaStatsTemp

    Try
        Call myManovaStatistics.manova_between(1, BetweenTemp, ✓
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon. ✓
Error)
            Return
        End Try
        Between(index) = BetweenTemp

    Try
        Call myManovaStatistics.manova_within(1, WithinTemp, ✓
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon. ✓
Error)
            Return
        End Try
        Within(index) = WithinTemp

    Try
        Call myManovaStatistics.manova_total(1, TotalTemp, SampleDataTestTempArray ✓
, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon. ✓
Error)
            Return
        End Try
        Total(index) = TotalTemp

    Try
        Call myManovaStatistics.manova_dfwithin(1, dfWithinTemp, ✓
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon. ✓
Error)
            Return
        End Try
        dfWithin(index) = dfWithinTemp

```

```

        Try
            Call myManovaStatistics.manova_dfbetween(1, dfBetweenTemp,
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        dfBetween(index) = dfBetweenTemp

        Try
            Call myManovaStatistics.manova_dftotal(1, dfTotalTemp,
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        dfTotal(index) = dfTotalTemp

        Try
            Call myManovaStatistics.manova_lambda(1, lambdaTemp,
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        lambda(index) = lambdaTemp

        Try
            Call myManovaStatistics.manova_chisq(1, chisqTemp, SampleDataTestTempArray
, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        chisq(index) = chisqTemp

        Try
            Call myManovaStatistics.manova_chisqdf(1, chisqdfTemp,
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        chisqdf(index) = chisqdfTemp

        Try
            Call myManovaStatistics.manova_eigenval(1, eigenvalTemp,
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try

```

```

        eigenval(index) = eigenvalTemp

        Try
            Call myManovaStatistics.manova_eigenvec(1, eigenvecTemp,
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        eigenvec(index) = eigenvecTemp

        Try
            Call myManovaStatistics.manova_canon(1, canonTemp, SampleDataTestTempArray
, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        canon(index) = canonTemp

        Try
            Call myManovaStatistics.manova_mdists(1, mdistsTemp, SampleDataTestTempArray
, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        mdists(index) = mdistsTemp

        Try
            Call myManovaStatistics.manova_gnames(1, gnamesTemp,
SampleDataTestTempArray, SampleNamesTestTempArray)
        Catch ex As Exception
            MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.
Error)
            Return
        End Try
        gnames(index) = gnamesTemp

    Next

    'Create a text file with the P values
    Dim testsString As String = ""
    Dim tempProbValue As Double
    Dim tempProbValueRound As Single
    Dim testStringTemp As String = ""
    Dim testsStringArray As Array
    For index = 0 To NumberOfTests - 1
        testsStringArray = CType(gnames.GetValue(index), Array)
        testsString = testsString + CType(testsStringArray.GetValue(1, 1), String) +
ControlChars.Tab + ControlChars.Tab
        testsString = testsString + CType(testsStringArray.GetValue(2, 1), String) +
ControlChars.Tab + ControlChars.Tab
        tempProbValue = CType(ManovaProb.GetValue(index), Double)
        If tempProbValue < 0.001 Then
            testStringTemp = "< 0.001"
        End If
        If tempProbValue < 0.01 And tempProbValue >= 0.001 Then
            testStringTemp = "< 0.01"
        End If
    Next

```

```

    If tempProbValue > 0.01 Then
        tempProbValueRound = CType(Math.Round(tempProbValue, 4), Single)
        testStringTemp = CType(tempProbValueRound, String)
    End If
    testsString = testsString + testStringTemp + ControlChars.Lf

Next

'Create the output text box
Dim richText As String = "Multiple analysis of variance completed successfully. "✓

-
+ ControlChars.Lf + ControlChars.Lf + _
    "In data sets with multiple variables, it is desireable to determine if the means ✓
of two samples are significantly different. A multiple analysis of variance (MANOVA) ✓
can be used to produce probability values. A P value of 0.01 essentially means that ✓
one can be 99% certain that chance alone would not lead to the differences seen ✓
between sample means. In this analysis, one must have a 'square' matrix. Therefore, ✓
the original data is expanded using a random number generator to produce the proper ✓
matrix dimensions. " _
+ ControlChars.Lf + ControlChars.Lf + _
    "The following table shows each test (Sample 1 vs Sample 2) and the P value. A P ✓
value of less than 0.05 indicates a significant difference." _
+ ControlChars.Lf + ControlChars.Lf + _
    "Sample 1" + ControlChars.Tab + ControlChars.Tab + "Sample 2" + ControlChars.Tab + ✓
ControlChars.Tab + "P value" _
+ ControlChars.Lf + _
    "-----" + ControlChars.Tab + ControlChars.Tab + "-----" + ControlChars.Tab + ✓
ControlChars.Tab + "-----" + ControlChars.Lf + ControlChars.Lf + testsString

Me.Between = Between
Me.Within = Within
Me.Total = Total
Me.dfWithin = dfWithin
Me.dfBetween = dfBetween
Me.dfTotal = dfTotal
Me.Lambda = lambda
Me.Chisq = chisq
Me.Eigenval = eigenval
Me.Eigenvec = eigenvec
Me.Canon = canon
Me.Mdist = mdist
Me.Gnames = gnames
Me.testsString = testsString
Me.richText = richText

End Sub

End Class

```

```
Imports System.Text.RegularExpressions
Imports Cl.Win.ClFlexGrid
```

```
Public Class pca
```

```
    Private myPCA_Output As PCA_output.PCA_output_data
    Private myNewVariances As Object
    Private mySamples As Integer
    Private myVariables As Integer
    Private mySelectedSamples() As Object
    Private myTempData As Object
    Private myRichText As String
```

```
    Public ReadOnly Property NewVariances() As Object
        Get
            Return myNewVariances
        End Get
    End Property
```

```
    Public ReadOnly Property Samples() As Integer
        Get
            Return mySamples
        End Get
    End Property
```

```
    Public ReadOnly Property Variables() As Integer
        Get
            Return myVariables
        End Get
    End Property
```

```
    Public ReadOnly Property SelectedSamples() As Object
        Get
            Return mySelectedSamples
        End Get
    End Property
```

```
    Public ReadOnly Property TempData() As Object
        Get
            Return myTempData
        End Get
    End Property
```

```
    Public ReadOnly Property RichText() As String
        Get
            Return myRichText
        End Get
    End Property
```

```
Friend Sub New(ByRef DataTable As Data_Table)
```

```
    Dim myPCA_Output As New PCA_output.PCA_output_data

    Dim i As Integer
    i = DataTable.DataTable.Rows.Count - 1
    'Define a grid with all of the data in column 1 and 2
    Dim SelectSamples As New Select_Samples
    Dim SampleNameList As New Cl.Win.ClFlexGrid.CellRange
    SampleNameList = DataTable.DataTable.GetCellRange(1, 1, i, 1)
    Dim newlineString As String = Nothing
    Dim Counter As Integer
    Dim sampleNameClip As String = SampleNameList.Clip
```

```

'Make the clip devoid of whitespace cells
For Counter = 1 To i
    If CType(DataTable.DataTable(Counter, 2), String) = "1" Then
        newlineString = newlineString & CType(DataTable.DataTable(Counter, 1),
String) & Environment.NewLine
    End If
Next

'Open up the Sample Selection Dialog
SelectSamples.samples = newlineString
SelectSamples.Text = "Choose Samples for PCA Analysis"
Try
    SelectSamples.ShowDialog()

Catch ex As Exception
    MessageBox.Show(ex.Message, "Error creating dialog", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)

End Try
If SelectSamples.DialogResult = DialogResult.Cancel Then
    Return
End If

'Determine which samples where selected and save the names in a String Clip
Dim SelectedSamples As String
SelectedSamples = SelectSamples.SampleChoice()
Dim q As Integer = 0
'Find where the alpha next to \n characters are
Dim re As New Regex("[a-zA-Z0-9]\x0D")
Dim mc As MatchCollection = re.Matches(SelectedSamples)
'Find out how many alpha or numbers next to \n characters there are
q = mc.Count

'Make an array of sample names displayed to user
Dim SelectedSampleArray(q, 1) As String
Dim SampleCounter As Integer
For SampleCounter = 0 To q
    SelectedSampleArray(SampleCounter, 0) = CType(SelectSamples.SelectSamples
(SampleCounter + 1, 1), String)
    SelectedSampleArray(SampleCounter, 1) = CType(SelectSamples.SelectSamples
(SampleCounter + 1, 2), String)

Next

'Make an array of selected samples to be processed
Dim SelectedSamplesUser(q) As String
Dim arraynumber As Integer = 0
For SampleCounter = 0 To q
    If SelectedSampleArray(SampleCounter, 0) = "True" Then
        SelectedSamplesUser(arraynumber) = SelectedSampleArray(SampleCounter, 1)
        arraynumber = arraynumber + 1
    End If
Next

'Check to make sure at least 2 samples were chosen
If arraynumber < 2 Then
    MessageBox.Show("You must select at least two samples", "Sample Selection",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    Return
End If

'Redimension the selected sample array
ReDim Preserve SelectedSamplesUser(arraynumber - 1)

'Find out how many columns have data in them
Dim NumberofVariables As CellRange

```

```

    NumberOfVariables = DataTable.DataTable.GetCellRange(1, 3, CType(DataTable.
DataTable.Rows.Count, Integer) - 1, CType(DataTable.DataTable.Cols.Count, Integer) -
1)
    Dim ColumnData, AdjacentColumnData As CellRange
    Dim l As Integer = 0
    Dim k, m As Integer

    'Count the number of filled in columns (i.e. how many variables).
    Dim rel As New Regex("[0-9]")
    For k = 3 To CType(DataTable.DataTable.Cols.Count, Integer) - 2
        ColumnData = DataTable.DataTable.GetCellRange(1, k, CType(DataTable.DataTable.
Rows.Count, Integer) - 1, k)
        'provide a counter to make sure all columns are contiguous
        AdjacentColumnData = DataTable.DataTable.GetCellRange(1, k + 1, CType
(DataTable.DataTable.Rows.Count, Integer) - 1, k + 1)
        If Not rel.Matches(ColumnData.Clip).Count = 0 Then
            l = l + 1
        End If
        'count if columns are not adjacent (i.e. any empty columns in between)
        If Not rel.Matches(ColumnData.Clip).Count = 0 And Not rel.Matches
(AdjacentColumnData.Clip).Count = 0 Then
            m = m + 1
        End If
    Next
    'Count last column if it has data in it
    k = k + 1
    If Not rel.Matches(ColumnData.Clip).Count = 0 Then
        l = l + 1
    End If
    'Make user reformat data so the routine will not break
    If Not m = l - 1 Then
        MessageBox.Show("It appears that you have a column with missing data. Please
delete or fill in any columns with no data that are inbetween data-bearing columns",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If

    'Determine the number of replicates
    Dim Replicates As Integer = CType(DataTable.Replicates, Integer)
    If Replicates = Nothing Then
        Dim ReplicateCells As CellRange
        ReplicateCells = DataTable.DataTable.GetCellRange(1, 2, CType(DataTable.
DataTable.Rows.Count - 1, Integer), 2)
        Dim maxReplicate As Integer
        maxReplicate = CType(DataTable.DataTable.Aggregate(AggregateEnum.Max,
ReplicateCells, AggregateFlags.None), Integer)
        Replicates = maxReplicate
    End If

    'Create an array to "hold" the averages and STDs of each group of data

    Dim x, z As Integer
    Dim AverageRange As CellRange
    Dim n As Integer = SelectedSamplesUser.GetLength(0)
    Dim SamplesToBeProcessed(n - 1) As String
    Dim PCAToBe(n - 1, 1 - 1) As Double
    Dim PCAStdToBe(n - 1, 1 - 1) As Double
    For m = 0 To n - 1
        SamplesToBeProcessed(m) = SelectedSamplesUser(m).ToString
        For k = 1 To CType(DataTable.DataTable.Rows.Count, Integer) - 1
            ColumnData = DataTable.DataTable.GetCellRange(k, 1, k, 1)
            If ColumnData.Clip = SelectedSamplesUser(m) Then
                For z = 3 To 2 + 1
                    AverageRange = DataTable.DataTable.GetCellRange(k, z, k +
Replicates - 1, z)

```

```

        PCAToBe(m, z - 3) = CType(DataTable.DataTable.Aggregate
(AggregateEnum.Average, AverageRange, AggregateFlags.None), Double)
        PCASTdsToBe(m, z - 3) = CType(DataTable.DataTable.Aggregate
(AggregateEnum.Std, AverageRange, AggregateFlags.None), Double)
    Next
    End If
Next
Next

Dim newdata As Object
Dim pcs As Object
Dim variances As Object
Dim t2 As Object
Try
    Call myPCA_Output.toms_pca_newdata(1, newdata, PCAToBe)
Catch ex As Exception
    'Will catch any error that we're not explicitly trapping.
    MessageBox.Show("Your Data Table has some problem with the 'newdata' routine.
Error message: " & ex.message, "Serious Data Problem", MessageBoxButtons.OK,
MessageBoxIcon.Stop)
    Return
End Try

Try
    Call myPCA_Output.toms_pca_pcs(1, pcs, PCAToBe)
Catch ex As Exception
    'Will catch any error that we're not explicitly trapping.
    MessageBox.Show("Your Data Table has some problem with the 'pcs' routine.
Error message: " & ex.message, "Serious Data Problem", MessageBoxButtons.OK,
MessageBoxIcon.Stop)
    Return
End Try

Try
    Call myPCA_Output.toms_pca_variances(1, variances, PCAToBe)
Catch ex As Exception
    'Will catch any error that we're not explicitly trapping.
    MessageBox.Show("Your Data Table has some problem 'variances' routine. Error
message: " & ex.Message, "Serious Data Problem", MessageBoxButtons.OK, MessageBoxIcon.
Stop)
    Return
End Try

Try
    Call myPCA_Output.toms_pca_t2(1, t2, PCAToBe)
Catch ex As Exception
    'Will catch any error that we're not explicitly trapping.
    MessageBox.Show("Your Data Table has some problem 't2' routine", "Serious Data
Problem", MessageBoxButtons.OK, MessageBoxIcon.Stop)
    Return
End Try

Dim tempnewdata As Array = CType(newdata, Array)
Dim newtempdata(n - 1, 1 - 1) As Object
For m = 0 To n - 1
    For k = 0 To 1 - 1
        newtempdata(m, k) = tempnewdata.GetValue(m + 1, k + 1)
    Next
Next

Dim tempnewvariances As Array = CType(variances, Array)
Dim newtempvariances(1 - 1) As Object
For k = 0 To 1 - 1
    newtempvariances(k) = tempnewvariances.GetValue(k + 1, 1)
Next

```

```

'Create a string of the pcs
Dim temppcs As Array = CType(pcs, Array)
Dim temppcsSingle As Single
Dim temppcsSingleRound As Single
Dim temppcstext As String
For m = 1 To temppcs.GetUpperBound(1)
    For k = 1 To 1
        temppcsSingle = CType(temppcs.GetValue(m, k), Single)
        temppcsSingleRound = CType(Math.Round(temppcsSingle, 4), Single)
        temppcstext = temppcstext & CType(temppcsSingleRound, String) +
ControlChars.Tab
    Next
    temppcstext = temppcstext + ControlChars.Lf
Next

'Create string of variances
Dim tempvariancesText As String
Dim tempVariance As Double
Dim tempVariancesSingle As Single = 0
Dim tempVariancesSingleRound As Single
For k = 0 To 1 - 1
    tempVariancesSingle = tempVariancesSingle + CType(newtempvariances(k), Single)
Next

For k = 0 To 1 - 1
    tempVariance = (100 * CType(newtempvariances(k), Single)) /
tempVariancesSingle
    tempVariancesSingleRound = CType(Math.Round(tempVariance, 4), Single)
    tempvariancesText = tempvariancesText & CType(tempVariancesSingleRound,
String) + ControlChars.Tab
Next

'Create a string of the newdata
Dim tempnewdataSingle As Single
Dim tempnewdataSingleRound As Single
Dim tempnewdatatext As String
For m = 1 To n - 1
    For k = 1 To 1
        tempnewdataSingle = CType(tempnewdata.GetValue(m, k), Single)
        tempnewdataSingleRound = CType(Math.Round(tempnewdataSingle, 5), Single)
        tempnewdatatext = tempnewdatatext & CType(tempnewdataSingleRound, String)
+ ControlChars.Tab
    Next
    tempnewdatatext = tempnewdatatext + ControlChars.Lf
Next

Dim richText As String = "PCA analysis completed successfully. " _
+ ControlChars.Lf + ControlChars.Lf + _
    "In data sets with multiple variables, groups of variables often behave similarly.
More than one variable may in fact be describing the same principle of the system.
PCA attempts to simplify a multivariate data set by replacing a group of variables
with a single new variable, called a principal component. Each principal component is
a linear combination of the original variables. The variance of each principal
component is the maximum among all possible choices. The analysis provides information
as to how much of the original variance is represented by each principal component.
Therefore, when the primary components are graphed against one-another, data sets that
are highly similar will plot together, while dissimilar data sets will occupy
different spaces on a graph." _
+ ControlChars.Lf + ControlChars.Lf + _
    "The result of placing the scores in a new coordinate system allows visualizing
the data. The loadings in the new coordinate system are shown here:" _
+ ControlChars.Lf + ControlChars.Lf + tempnewdatatext + ControlChars.Lf +
ControlChars.Lf + _
    "Variance data are also produced during a PCA. Each variance output corresponds

```

to the principal components loadings shown above (by column). The variances (as %) are shown in the bar chart (with cumulative totals shown in the bar labels) and below: ✓

" - + ControlChars.Lf + ControlChars.Lf + tempvariancesText + ControlChars.Lf + ControlChars.Lf + _ ✓

"The whole purpose of PCA is to reduce the majority of the variability to just a few new variables. For this reason, the first and second new variables are plotted against one another. Most of the variability is depicted in 'Component One' (shown graphically on the X-Axis of the scatter plot). Variability in 'Component Two' (Y-Axis of scatter plot) provides a second dimension with the second-most variability. Therefore where samples cluster, they are more closely related." ✓

```
Me.myNewVariances = newtempvariances
Me.mySamples = n
Me.myVariables = 1
Me.mySelectedSamples = SelectedSamplesUser
Me.myTempData = newtempdata
Me.myRichText = richText
```

End Sub

End Class

```
Imports Cl.Win.ClChart
Imports System.Drawing.Imaging
Imports System.Drawing.Printing
Imports Cl.ClPrintDocument
```

```
Public Class Plot
    Inherits System.Windows.Forms.Form
```

```
#Region " Windows Form Designer generated code "
```

```
    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
```

```
End Sub
```

```
'Form overrides dispose to clean up the component list.
Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
        If Not (components Is Nothing) Then
            components.Dispose()
        End If
    End If
    MyBase.Dispose(disposing)
End Sub
```

```
'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer
```

```
'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents chartPCA As Cl.Win.ClChart.ClChart
Friend WithEvents ctxCopy As System.Windows.Forms.MenuItem
Friend WithEvents ctxSaveAs As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem
Friend WithEvents ctxPrint As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem6 As System.Windows.Forms.MenuItem
Friend WithEvents ctxExit As System.Windows.Forms.MenuItem
Friend WithEvents ContextMenuPlot As System.Windows.Forms.ContextMenu
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Dim resources As System.Resources.ResourceManager = New System.Resources.
ResourceManager(GetType(Plot))
    Me.chartPCA = New Cl.Win.ClChart.ClChart
    Me.ContextMenuPlot = New System.Windows.Forms.ContextMenu
    Me.ctxCopy = New System.Windows.Forms.MenuItem
    Me.ctxSaveAs = New System.Windows.Forms.MenuItem
    Me.MenuItem3 = New System.Windows.Forms.MenuItem
    Me.ctxPrint = New System.Windows.Forms.MenuItem
    Me.MenuItem6 = New System.Windows.Forms.MenuItem
    Me.ctxExit = New System.Windows.Forms.MenuItem
    CType(Me.chartPCA, System.ComponentModel.ISupportInitialize).BeginInit()
    Me.SuspendLayout()
    '
    'chartPCA
    '
    Me.chartPCA.BackColor = System.Drawing.Color.White
    Me.chartPCA.DataSource = Nothing
    Me.chartPCA.Dock = System.Windows.Forms.DockStyle.Fill
    Me.chartPCA.Location = New System.Drawing.Point(0, 0)
    Me.chartPCA.Name = "chartPCA"
    Me.chartPCA.PropBag = "<?xml version='1.0'?'><Chart2DPropBag Version=''"
```

```

<StyleCollection><NamedStyle><ParentName><StyleData>Border=None,Black,1;</StyleData><Name>PlotAr" &
    "entName>Area</ParentName><StyleData>Border=None,Black,1;</StyleData><Name>PlotAr" &
    &
    "ea</Name></NamedStyle><NamedStyle><ParentName>Legend.default</ParentName><StyleD" &
    &
    "ata>BackColor=Window;</StyleData><Name>Legend</Name></NamedStyle><NamedStyle><Pa" &
    &
    "rentName>Control</ParentName><StyleData>Border=None,Black,1;</StyleData><Name>Fo" &
    &
    "oter</Name></NamedStyle><NamedStyle><ParentName>Area.default</ParentName><StyleD" &
    &
    "ata /><Name>Area</Name></NamedStyle><NamedStyle><ParentName>Control.default</Par" &
    &
    "entName><StyleData>BackColor=White;</StyleData><Name>Control</Name></NamedStyle>" &
    &
    "<NamedStyle><ParentName>Area</ParentName><StyleData>Rotation=Rotate0;Border=None" &
    &
    ",Transparent,1;AlignHorz=Center;BackColor=Transparent;Opaque=False;Font=Microsof" &
    &
    "t Sans Serif, 8.25pt;AlignVert=Bottom;</StyleData><Name>AxisX</Name></NamedStyle" &
    &
    "><NamedStyle><ParentName>Area</ParentName><StyleData>Rotation=Rotate270;Border=N" &
    &
    "one,Transparent,1;AlignHorz=Near;BackColor=Transparent;Opaque=False;Font=Microso" &
    &
    "ft Sans Serif, 8.25pt;AlignVert=Center;</StyleData><Name>AxisY</Name></NamedStyl" &
    &
    "e><NamedStyle><ParentName>LabelStyleDefault.default</ParentName><StyleData /><Na" &
    &
    "me>LabelStyleDefault</Name></NamedStyle><NamedStyle><ParentName>Control</ParentN" &
    &
    "ame><StyleData>Border=None,Black,1;Wrap=False;AlignVert=Top;</StyleData><Name>Le" &
    &
    "gend.default</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><Sty" &
    &
    "leData>Border=None,Black,1;BackColor=Transparent;</StyleData><Name>LabelStyleDef" &
    &
    "ault.default</Name></NamedStyle><NamedStyle><ParentName>Control</ParentName><Sty" &
    &
    "leData>Border=None,Black,1;BackColor2=White;BackColor=White;</StyleData><Name>He" &
    &
    "ader</Name></NamedStyle><NamedStyle><ParentName /><StyleData>ForeColor=ControlTe" &
    &
    "xt;Border=None,Black,1;BackColor=Control;</StyleData><Name>Control.default</Name" &
    &
    "></NamedStyle><NamedStyle><ParentName>Area</ParentName><StyleData>Rotation=Rotat" &
    &
    "e90;Border=None,Transparent,1;AlignHorz=Far;BackColor=Transparent;AlignVert=Cent" &
    &
    "er;</StyleData><Name>AxisY2</Name></NamedStyle><NamedStyle><ParentName>Control</" &
    &
    "ParentName><StyleData>Border=None,Black,1;AlignVert=Top;</StyleData><Name>Area.d" &
    &
    "efault</Name></NamedStyle></StyleCollection><Header Compass="North"><Text>
Princi" &
    "pal Components Loadings</Text></Header><Footer Compass="South"><Text /></
Footer>" &
    "<Legend Visible="False" Compass="East"><Text /></Legend><ChartArea /><Axes>
<Axis" &
    " UnitMajor="0.2" UnitMinor="0.1" AutoMajor="True" AutoMinor="True"
AutoMax="True" &
    "" AutoMin="True" Max="1" Min="-1" _onTop="0" Compass="South">
<GridMajor AutoSpac" &
    "e="True" Color="LightGray" Pattern="Dash" Thickness="1" /><GridMinor
AutoSpace="" &
    "True" Color="LightGray" Pattern="Dash" Thickness="1" /><Text /></Axis>
<Axis Unit" &

```

```

        "Major="0.2" UnitMinor="0.1" AutoMajor="True" AutoMinor="True" AutoMax="
"True" Aut" & _
        "oMin="True" Max="1" Min="-1" _onTop="0" Compass="West"><GridMajor
AutoSpace="Tru" & _
        "e" Color="LightGray" Pattern="Dash" Thickness="1" /><GridMinor AutoSpace=
"True" " & _
        "Color="LightGray" Pattern="Dash" Thickness="1" /><Text /></Axis><Axis
UnitMajor=" & _
        "0" UnitMinor="0" AutoMajor="True" AutoMinor="True" AutoMax="True"
AutoMin="True" & _
        "Max="0" Min="0" _onTop="0" Compass="East"><GridMajor AutoSpace="True
" Color="L" & _
        "ightGray" Pattern="Dash" Thickness="1" /><GridMinor AutoSpace="True" Color
="Ligh" & _
        "tGray" Pattern="Dash" Thickness="1" /><Text /></Axis></Axes>
<ChartGroupsCollecti" & _
        "on"><ChartGroup><ShowOutline>True</ShowOutline><HiLoData>FillFalling=True, FillTra
& _
        "nsparent=True, FullWidth=False, ShowClose=True, ShowOpen=True</HiLoData><ChartType>"
& _
        "XYPlot</ChartType><Name>Group1</Name><Bar>ClusterOverlap=0, ClusterWidth=50</Bar>"
& _
        "<DataSerializer Hole="3.4028234663852886E+38" DefaultSet="True">
<DataSeriesColle" & _
        "ction"><DataSeriesSerializer><SeriesLabel>Component One</SeriesLabel><DataTypes>D"
& _
        "ouble;Double;Double;Double;Double</DataTypes><DataFields>;;/DataFields><Symbo"
& _
        "lStyle Size="8" Color="Black" Shape="Square" /><X /><Y1 /><Y /><LineStyle
Color=" & _
        "DarkKhaki" Pattern="None" Thickness="1" /><Tag /><Y2 /><Y3 /></
DataSeriesSerial" & _
        "izer"><DataSeriesSerializer><SeriesLabel>Component Two</SeriesLabel><DataTypes>Do"
& _
        "uble;Double;Double;Double;Double</DataTypes><DataFields>;;/DataFields><Symbol"
& _
        "Style Size="8" Color="Black" Shape="Square" /><X /><Y1 /><Y /><LineStyle
Color=" & _
        "DarkMagenta" Pattern="None" Thickness="1" /><Tag /><Y2 /><Y3 /></
DataSeriesSeria" & _
        "lizer"></DataSeriesCollection></DataSerializer><Bubble>EncodingMethod=Diameter, Ma
& _
        "ximumSize=20, MinimumSize=5</Bubble><Pie>OtherOffset=0, Start=0</Pie><Polar>Degree"
& _
        "s=True, PiRatioAnnotations=True, Start=0</Polar><Stacked>False</Stacked><Radar>Deg"
& _
        "rees=True, Filled=False, Start=0</Radar><Visible>True</Visible></ChartGroup><Chart"
& _
        "Group><ShowOutline>True</ShowOutline><HiLoData>FillFalling=True, FillTransparent="
& _
        "True, FullWidth=False, ShowClose=True, ShowOpen=True</HiLoData><ChartType>XYPlot</C"
& _
        "hartType><Name>Group2</Name><Bar>ClusterOverlap=0, ClusterWidth=50</Bar><DataSeri"
& _
        "alizer Hole="3.4028234663852886E+38" /><Bubble>EncodingMethod=Diameter,
MaximumSi" & _
        "ze=20, MinimumSize=5</Bubble><Pie>OtherOffset=0, Start=0</Pie><Polar>Degrees=True, "
& _
        "PiRatioAnnotations=True, Start=0</Polar><Stacked>False</Stacked><Radar>Degrees=Tr"
& _
        "ue, Filled=False, Start=0</Radar><Visible>True</Visible></ChartGroup></ChartGroups"
& _
        "Collection></Chart2DPropBag>"
Me.chartPCA.Size = New System.Drawing.Size(422, 373)
Me.chartPCA.TabIndex = 0
'ContextMenuPlot

```

```

    Me.ContextMenuPlot.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
ctxCopy, Me.ctxSaveAs, Me.MenuItem3, Me.ctxPrint, Me.MenuItem6, Me.ctxExit})
    'ctxCopy
    '
    Me.ctxCopy.Index = 0
    Me.ctxCopy.Text = "&Copy"
    '
    'ctxSaveAs
    '
    Me.ctxSaveAs.Index = 1
    Me.ctxSaveAs.Text = "Save &As"
    '
    'MenuItem3
    '
    Me.MenuItem3.Index = 2
    Me.MenuItem3.Text = "- "
    '
    'ctxPrint
    '
    Me.ctxPrint.Index = 3
    Me.ctxPrint.Text = "&Print"
    '
    'MenuItem6
    '
    Me.MenuItem6.Index = 4
    Me.MenuItem6.Text = "- "
    '
    'ctxExit
    '
    Me.ctxExit.Index = 5
    Me.ctxExit.Text = "E&xit"
    '
    'Plot
    '
    Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
    Me.ClientSize = New System.Drawing.Size(422, 373)
    Me.ContextMenu = Me.ContextMenuPlot
    Me.Controls.Add(Me.chartPCA)
    Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
    Me.Name = "Plot"
    Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
    Me.Text = "Plot"
    CType(Me.chartPCA, System.ComponentModel.ISupportInitialize).EndInit()
    Me.ResumeLayout(False)

```

End Sub

#End Region

```

Private mySamples As Integer
Private myVariables As Integer
Private myInput_data As Array
Private mySampleNames As Array

Public Property Variables() As Integer
    Get
        Return myVariables
    End Get
    Set(ByVal Value As Integer)
        myVariables = Value
    End Set
End Property

Public Property Samples() As Integer
    Get

```

```

        Return mySamples
    End Get
    Set(ByVal Value As Integer)
        mySamples = Value
    End Set
End Property

Public Property Input_data() As Array
    Get
        Return myInput_data
    End Get
    Set(ByVal Value As Array)
        myInput_data = Value
    End Set
End Property

Public Property SampleNames() As Array
    Get
        Return mySampleNames
    End Get
    Set(ByVal Value As Array)
        mySampleNames = Value
    End Set
End Property

Private Sub chartPCA_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles chartPCA.Load

    Dim chartData As Cl.Win.ClChart.ChartDataSeries
    Dim chartDataXY As Cl.Win.ClChart.ChartData
    Dim chartLabels As Cl.Win.ClChart.ChartLabels
    Dim chartLabel As Label
    Dim AxisCounter As Integer
    Dim xAxisData(Samples - 1) As Double
    Dim yAxisData(Samples - 1) As Double
    chartPCA.Style.Border.BorderStyle = Cl.Win.ClChart.BorderStyleEnum.Solid
    chartPCA.Style.Border.Thickness = 1

    For AxisCounter = 0 To Samples - 1
        xAxisData(AxisCounter) = CType(Input_data.GetValue(AxisCounter, 0), Double)
        yAxisData(AxisCounter) = CType(Input_data.GetValue(AxisCounter, 1), Double)
    Next

    chartPCA.ChartGroups(0).ChartData.SeriesList(0).X.CopyDataIn(xAxisData)
    chartPCA.ChartGroups(0).ChartData.SeriesList(0).Y.CopyDataIn(yAxisData)
    chartPCA.ChartArea.AxisX.Text = ControlChars.Lf + "Component One"
    chartPCA.ChartArea.AxisY.Text = "Component Two" + ControlChars.Lf + "    "
    chartPCA.Header.Style.Font = New Font("Arial", 10, FontStyle.Bold)

    'Add +- Lines
    Dim xLowerBound As Integer = xAxisData.GetLowerBound(0)
    Dim xUpperBound As Integer = xAxisData.GetUpperBound(0)
    Dim xMin As Double
    Dim xMax As Double
    Dim maxIndex As Integer = 0
    Dim minIndex As Integer = 0
    Dim yLowerBound As Integer = yAxisData.GetLowerBound(0)
    Dim yUpperBound As Integer = yAxisData.GetUpperBound(0)
    Dim yMin As Double
    Dim yMax As Double
    Dim Counter As Integer

    'Find maximum X value
    xMax = xAxisData(0)
    For Counter = xLowerBound To xUpperBound
        If xAxisData(maxIndex) < xAxisData(Counter) Then
            maxIndex = Counter
        End If
    Next

```

```

        xMax = xAxisData(maxIndex)
    End If
Next

'Find minimum X value
xMin = xAxisData(0)
For Counter = xLowerBound + 1 To xUpperBound
    If xMin > xAxisData(Counter) Then
        minIndex = Counter
        xMin = xAxisData(minIndex)
    End If
Next

'Find maximum Y value
maxIndex = 0
For Counter = yLowerBound To yUpperBound
    If yAxisData(maxIndex) < yAxisData(Counter) Then
        maxIndex = Counter
        yMax = yAxisData(maxIndex)
    End If
Next

'Find minimum Y value
yMin = yAxisData(0)
For Counter = yLowerBound + 1 To yUpperBound
    If yMin > yAxisData(Counter) Then
        minIndex = Counter
        yMin = yAxisData(minIndex)
    End If
Next

Dim xMinMod As Integer = CType(xMin, Integer)
Dim xMaxMod As Integer = CType(xMax, Integer)
If xMinMod > xMin Then
    xMinMod = xMinMod - 1
End If
If xMaxMod < xMax Then
    xMaxMod = xMaxMod + 1
End If

Dim yMinMod As Integer = CType(yMin, Integer)
Dim yMaxMod As Integer = CType(yMax, Integer)
If yMinMod > yMin Then
    yMinMod = yMinMod - 1
End If
If yMaxMod < yMax Then
    yMaxMod = yMaxMod + 1
End If

xMinMod = xMinMod - 1
xMaxMod = xMaxMod + 1
yMinMod = yMinMod - 1
yMaxMod = yMaxMod + 1

chartPCA.ChartArea.AxisX.Min = xMinMod
chartPCA.ChartArea.AxisX.Max = xMaxMod
chartPCA.ChartArea.AxisY.Min = yMinMod
chartPCA.ChartArea.AxisY.Max = yMaxMod

'Enter data into plot
Dim group1 As Cl.Win.ClChart.ChartGroup = chartPCA.ChartGroups(0)
group1.ChartType = Cl.Win.ClChart.Chart2DTypeEnum.XYPlot
group1.DrawingOrder = 0

Dim Horizdata As Cl.Win.ClChart.ChartData = group1.ChartData

```

```

Dim sHoriz As New Cl.Win.ClChart.ChartDataSeries
Horizdata.SeriesList.Add(sHoriz)
sHoriz.FitType = Cl.Win.ClChart.FitTypeEnum.Line
sHoriz.SymbolStyle.Shape = Cl.Win.ClChart.SymbolShapeEnum.None
sHoriz.LineStyle.Color = Color.Black
Dim pfa() As PointF = {New PointF(xMinMod, 0.0F), New PointF(xMaxMod, 0.0F)}
sHoriz.PointData.CopyDataIn(pfa)

'Enter data into plot
Dim group2 As Cl.Win.ClChart.ChartGroup = chartPCA.ChartGroups(1)
group2.ChartType = Cl.Win.ClChart.Chart2DTypeEnum.XYPlot
group2.DrawingOrder = 1
Dim Verticaldata As Cl.Win.ClChart.ChartData = group2.ChartData

Dim sVert As New Cl.Win.ClChart.ChartDataSeries
Verticaldata.SeriesList.Add(sVert)
sVert.FitType = Cl.Win.ClChart.FitTypeEnum.Line
sVert.SymbolStyle.Shape = Cl.Win.ClChart.SymbolShapeEnum.None
sVert.LineStyle.Color = Color.Black
Dim pfal() As PointF = {New PointF(0.0F, yMinMod), New PointF(0.0F, yMaxMod)}
sVert.PointData.CopyDataIn(pfal)
chartPCA.ChartArea.AxisY2.Visible = False

'Make box around graph
sVert = New ChartDataSeries
Verticaldata.SeriesList.Add(sVert)
pfal = New PointF() {New PointF(xMaxMod, yMinMod), New PointF(xMaxMod, yMaxMod)}
sVert.PointData.CopyDataIn(pfal)
sVert.SymbolStyle.Shape = Cl.Win.ClChart.SymbolShapeEnum.None
sVert.LineStyle.Color = Color.Black
sVert.LineStyle.Thickness = 2

sHoriz = New ChartDataSeries
Horizdata.SeriesList.Add(sHoriz)
pfal = New PointF() {New PointF(xMinMod, yMaxMod), New PointF(xMaxMod, yMaxMod)}
sHoriz.PointData.CopyDataIn(pfal)
sHoriz.SymbolStyle.Shape = Cl.Win.ClChart.SymbolShapeEnum.None
sHoriz.LineStyle.Color = Color.Black
sHoriz.LineStyle.Thickness = 3

'Add Sample Names to Labels Collection
Dim cLabs As ChartLabels = chartPCA.ChartLabels
cLabs.DefaultLabelStyle.BackColor = Color.Transparent
cLabs.DefaultLabelStyle.Border.BorderStyle = BorderStyleEnum.Empty
cLabs.DefaultLabelStyle.Border.Thickness = 0
Dim MyValue(SampleNames.GetUpperBound(0)) As Integer

For Counter = 0 To SampleNames.GetUpperBound(0)
    'Make a random number for the direction and distance offset.

    Randomize() ' Initialize random-number generator.
    MyValue(Counter) = CInt(Int((10 * Rnd()) + 1)) ' Generate random value between
1 and 6.
    Dim cLab As Cl.Win.ClChart.Label = cLabs.LabelsCollection.AddNewLabel()
    cLab.Text = SampleNames.GetValue(Counter).ToString
    cLab.AttachMethod = AttachMethodEnum.DataIndex
    cLab.AttachMethodData.GroupIndex = 0
    cLab.AttachMethodData.SeriesIndex = 0
    cLab.AttachMethodData.PointIndex = Counter
    cLab.Connected = True
    cLab.Offset = (MyValue(Counter) * 4)
    cLab.Visible = True
    cLab.Compass = LabelCompassEnum.Orthogonal
    'cLab.Compass = CType(MyValue(Counter), LabelCompassEnum)
    'If xAxisData(Counter) > 0 Then
    '    cLab.Compass = LabelCompassEnum.NorthWest

```

```

        'Else
        '    cLab.Compass = LabelCompassEnum.NorthEast
        'End If
    Next

End Sub

Private Sub mnuClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Me.Close()
End Sub

Private Sub ctxSaveAs_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles ctxSaveAs.Click
    Dim lastFilterIndex As Integer = 1
    Dim myPlot As Plot = Me
    Dim sfg As New SaveFileDialog

    sfg.Filter = "Metafiles (*.emf)|*.emf|" + "Bmp files (*.bmp)|*.bmp|" + "Gif files
(*.gif)|*.gif|" + "Jpeg files (*.jpg;*.jpeg)|*.jpg;*.jpeg|" + "Png files (*.png)|*.png
|" + "All graphic files (*.emf;*.bmp;*.gif;*.jpg;*.jpeg;*.png)|*.emf;*.bmp;*.gif;*.jpg
;*.jpeg;*.png"
    sfg.FilterIndex = lastFilterIndex
    sfg.OverwritePrompt = True
    sfg.CheckPathExists = True
    sfg.RestoreDirectory = False
    sfg.ValidateNames = True

    If sfg.ShowDialog() = DialogResult.OK Then
        Dim fn As String = sfg.FileName
        Dim indext As Integer = fn.LastIndexOf(".c")
        If indext < 0 Then
            indext = fn.Length + 1
            fn += ".emf"
        Else
            indext += 1
        End If
        Dim ext As String = fn.Substring(indext)
        Dim imgfmt As ImageFormat = Nothing

        Select Case ext
            Case "emf"
                imgfmt = ImageFormat.Emf
                myPlot.chartPCA.SaveImage(fn, imgfmt)

            Case "bmp"
                imgfmt = ImageFormat.Bmp

            Case "gif"
                imgfmt = ImageFormat.Gif

            Case "jpeg", "jpg"
                imgfmt = ImageFormat.Jpeg

            Case "png"
                imgfmt = ImageFormat.Png

            Case Else
                Return
        End Select

        lastFilterIndex = sfg.FilterIndex

        If Not imgfmt.Equals(ImageFormat.Emf) Then
            Dim img As Image = myPlot.chartPCA.GetImage()
            img.Save(fn, imgfmt)
            img.Dispose()
        End If
    End If
End Sub

```

```

        End If
    End If
    sfg.Dispose()
End Sub

Private Sub ctxCopy_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles ctxCopy.Click
    Dim myPlot As Plot = Me
    myPlot.chartPCA.SaveImage(ImageFormat.Emf)

End Sub

Private Sub ctxExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles ctxExit.Click
    Me.Close()
End Sub

Private Sub chartPCA_Click(ByVal sender As Object, ByVal e As System.EventArgs) ✓
Handles chartPCA.Click
    Me.Activate()
End Sub

Private Sub ctxPrint_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles ctxPrint.Click
    Dim doc As New C1PrintDocument
    Doc2D_PCA(doc, New GenerateEventArgs)
    Dim aprev As New Final_Report
    AddHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_PCA)
    aprev.C1PrintPreview1.Document = doc
    aprev.ShowDialog()
    RemoveHandler doc.GenerateDocument, New GenerateEventHandler(AddressOf Doc2D_PCA)
    aprev.Dispose()
    'barChart.chartBar.PrintChart(PrintScaleEnum.ScaleToFit)

End Sub

Private Sub Doc2D_PCA(ByVal doc As C1PrintDocument, ByVal e As GenerateEventArgs)
    Dim C1Chart1Raw As Plot = Me
    Dim C1Chart1 As C1.Win.C1Chart.C1Chart = C1Chart1Raw.chartPCA
    With doc
        .DefaultUnit = UnitTypeEnum.Mm
        .StartDoc()
        '.RenderBlockText("Chart", 50, 50, Nothing)
        Dim ww As Double = (CType(.BodyAreaSize.Width, Double)) * 0.9
        .RenderBlockC1Printable(C1Chart1, (.BodyAreaSize.Width * 0.9))
        .CanChangePageMetrics()
        .RenderBlockGraphicsBegin()
        .EndDoc()
    End With
End Sub

End Class

```

```

Public Class Properties
    Inherits System.Windows.Forms.Form

    #Region " Windows Form Designer generated code "

        Public Sub New()
            MyBase.New()

            'This call is required by the Windows Form Designer.
            InitializeComponent()

            'Add any initialization after the InitializeComponent() call

        End Sub

        'Form overrides dispose to clean up the component list.
        Protected Overrides Sub Dispose(ByVal disposing As Boolean)
            If disposing Then
                If Not (components Is Nothing) Then
                    components.Dispose()
                End If
            End If
            MyBase.Dispose(disposing)
        End Sub

        'Required by the Windows Form Designer
        Private components As System.ComponentModel.IContainer

        'NOTE: The following procedure is required by the Windows Form Designer
        'It can be modified using the Windows Form Designer.
        'Do not modify it using the code editor.
        Friend WithEvents Label1 As System.Windows.Forms.Label
        Friend WithEvents btnOK As System.Windows.Forms.Button
        <System.Diagnostics.DebuggerStepThrough() Private Sub InitializeComponent()
            Dim resources As System.Resources.ResourceManager = New System.Resources.
ResourceManager(GetType(Properties))
            Me.Label1 = New System.Windows.Forms.Label
            Me.btnOK = New System.Windows.Forms.Button
            Me.SuspendLayout()
            '
            'Label1
            '
            Me.Label1.Font = New System.Drawing.Font("Microsoft Sans Serif", 10.0!, System.
Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
            Me.Label1.Location = New System.Drawing.Point(16, 40)
            Me.Label1.Name = "Label1"
            Me.Label1.Size = New System.Drawing.Size(264, 160)
            Me.Label1.TabIndex = 0
            Me.Label1.Text = "There are currently no user settable properties for this
application"
            '
            'btnOK
            '
            Me.btnOK.DialogResult = System.Windows.Forms.DialogResult.Cancel
            Me.btnOK.Location = New System.Drawing.Point(105, 208)
            Me.btnOK.Name = "btnOK"
            Me.btnOK.TabIndex = 1
            Me.btnOK.Text = "&OK"
            '
            'Properties
            '
            Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
            Me.ClientSize = New System.Drawing.Size(292, 273)
            Me.ControlBox = False
            Me.Controls.Add(Me.btnOK)
            Me.Controls.Add(Me.Label1)
            Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        End Sub
    End Class

```

```
Me.MaximizeBox = False
Me.Name = "Properties"
Me.Text = "Properties"
Me.ResumeLayout(False)
```

```
End Sub
```

```
#End Region
```

```
Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles btnOK.Click
    Me.Close()
End Sub
```

```
End Class
```

```

Imports Cl.Win.C1FlexGrid
Imports System.Text.RegularExpressions

Public Class Select_Samples
    Inherits System.Windows.Forms.Form

    #Region " Windows Form Designer generated code "

        Public Sub New()
            MyBase.New()

            'This call is required by the Windows Form Designer.
            InitializeComponent()

            'Add any initialization after the InitializeComponent() call.

        End Sub

        'Form overrides dispose to clean up the component list.
        Protected Overrides Sub Dispose(ByVal disposing As Boolean)
            If disposing Then
                If Not (components Is Nothing) Then
                    components.Dispose()
                End If
            End If
            MyBase.Dispose(disposing)
        End Sub

        'Required by the Windows Form Designer
        Private components As System.ComponentModel.IContainer

        'NOTE: The following procedure is required by the Windows Form Designer
        'It can be modified using the Windows Form Designer.
        'Do not modify it using the code editor.
        Friend WithEvents btnSelectAll As System.Windows.Forms.Button
        Friend WithEvents btnSelectNone As System.Windows.Forms.Button
        Friend WithEvents btnSelectSamplesOK As System.Windows.Forms.Button
        Friend WithEvents btnSelectSamplesCancel As System.Windows.Forms.Button
        Friend WithEvents SelectSamples As Cl.Win.C1FlexGrid.C1FlexGrid
        <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
            Dim resources As System.Resources.ResourceManager = New System.Resources.
ResourceManager(GetType(Select_Samples))
            Me.SelectSamples = New Cl.Win.C1FlexGrid.C1FlexGrid
            Me.btnSelectAll = New System.Windows.Forms.Button
            Me.btnSelectNone = New System.Windows.Forms.Button
            Me.btnSelectSamplesOK = New System.Windows.Forms.Button
            Me.btnSelectSamplesCancel = New System.Windows.Forms.Button
            CType(Me.SelectSamples, System.ComponentModel.ISupportInitialize).BeginInit()
            Me.SuspendLayout()
            '
            'SelectSamples
            '
            Me.SelectSamples.AllowSorting = Cl.Win.C1FlexGrid.AllowSortingEnum.None
            Me.SelectSamples.BackColor = System.Drawing.SystemColors.Window
            Me.SelectSamples.ColumnInfo = "3,0,0,0,0,85,Columns:0{Visible:False;}" & Microsoft
.VisualBasic.ChrW(9) & "1{Width:37;AllowSorting:False;AllowDraggin" &
            "g:False;AllowResizing:False;DataType:System.Boolean;ImageAlign:CenterCenter;}" &
Microsoft.VisualBasic.ChrW(9) & "2{" &
            "Width:175;Caption: ""Sample"";AllowDragging:False;AllowResizing:False;AllowMerging
:" &
            "True;AllowEditing:False;TextAlign:LeftCenter;TextAlignFixed:CenterCenter;}" &
Microsoft.VisualBasic.ChrW(9)
            Me.SelectSamples.ForeColor = System.Drawing.SystemColors.WindowText
            Me.SelectSamples.Location = New System.Drawing.Point(16, 16)
            Me.SelectSamples.Name = "SelectSamples"
            Me.SelectSamples.Rows.Count = 750
            Me.SelectSamples.Size = New System.Drawing.Size(232, 440)

```

```

Me.SelectSamples.Styles = New Cl.Win.ClFlexGrid.CellStyleCollection("Fixed
{BackColor:Control;ForeColor:ControlText;Border:Flat,1,ControlDark,Both;}" & Microsoft
.VisualBasic.ChrW(9) & "Hi" &
"ghlight{BackColor:Highlight;ForeColor:HighlightText;}" & Microsoft.VisualBasic.
ChrW(9) & "Search{BackColor:Highlight" &
";ForeColor:HighlightText;}" & Microsoft.VisualBasic.ChrW(9) & "Frozen{BackColor:
Beige;}" & Microsoft.VisualBasic.ChrW(9) & "EmptyArea{BackColor:AppWorks" &
"pace;Border:Flat,1,ControlDarkDark,Both;}" & Microsoft.VisualBasic.ChrW(9) &
"GrandTotal{BackColor:Black;ForeColor:W" &
"hite;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal0{BackColor:ControlDarkDark;
ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal1{BackColor" &
":ControlDarkDark;ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal2
{BackColor:ControlDarkDark;ForeColor" &
":White;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal3{BackColor:ControlDarkDark;
ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal4{BackCol" &
"or:ControlDarkDark;ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) &
"Subtotal5{BackColor:ControlDarkDark;ForeCol" &
"or:White;}" & Microsoft.VisualBasic.ChrW(9))
Me.SelectSamples.TabIndex = 0
,
'btnSelectAll
,
Me.btnSelectAll.Location = New System.Drawing.Point(264, 32)
Me.btnSelectAll.Name = "btnSelectAll"
Me.btnSelectAll.TabIndex = 1
Me.btnSelectAll.Text = "Select &All"
,
'btnSelectNone
,
Me.btnSelectNone.Location = New System.Drawing.Point(264, 88)
Me.btnSelectNone.Name = "btnSelectNone"
Me.btnSelectNone.TabIndex = 2
Me.btnSelectNone.Text = "Select &None"
,
'btnSelectSamplesOK
,
Me.btnSelectSamplesOK.DialogResult = System.Windows.Forms.DialogResult.OK
Me.btnSelectSamplesOK.Location = New System.Drawing.Point(264, 360)
Me.btnSelectSamplesOK.Name = "btnSelectSamplesOK"
Me.btnSelectSamplesOK.TabIndex = 3
Me.btnSelectSamplesOK.Text = "&OK"
,
'btnSelectSamplesCancel
,
Me.btnSelectSamplesCancel.DialogResult = System.Windows.Forms.DialogResult.Cancel
Me.btnSelectSamplesCancel.Location = New System.Drawing.Point(264, 416)
Me.btnSelectSamplesCancel.Name = "btnSelectSamplesCancel"
Me.btnSelectSamplesCancel.TabIndex = 4
Me.btnSelectSamplesCancel.Text = "&Cancel"
,
'Select_Samples
,
Me.AcceptButton = Me.btnSelectSamplesOK
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.CancelButton = Me.btnSelectSamplesCancel
Me.ClientSize = New System.Drawing.Size(360, 469)
Me.ControlBox = False
Me.Controls.Add(Me.btnSelectSamplesCancel)
Me.Controls.Add(Me.btnSelectSamplesOK)
Me.Controls.Add(Me.btnSelectNone)
Me.Controls.Add(Me.btnSelectAll)
Me.Controls.Add(Me.SelectSamples)
Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
Me.MaximizeBox = False
Me.MinimizeBox = False
Me.Name = "Select_Samples"

```

```

Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
Me.Text = "SelectSamples"
CType(Me.SelectSamples, System.ComponentModel.ISupportInitialize).EndInit()
Me.ResumeLayout(False)

```

```
End Sub
```

```
#End Region
```

```

Private mSampleChoice As String
Public Property SampleChoice() As String
    Get
        Return CType(mSampleChoice, String)
    End Get
    Set(ByVal Value As String)
        mSampleChoice = Value
    End Set
End Property

```

```

Private mSamples As String
Public Property samples() As String
    Get
        Return CType(mSamples, String)
    End Get
    Set(ByVal Value As String)
        mSamples = Value
    End Set
End Property

```

```
Private Sub SelectSamples_Load(ByVal sender As System.Object, ByVal e As System. ✓
EventArgs) Handles MyBase.Load
```

```

    Dim sampleNamesCellRange As CellRange
    sampleNamesCellRange = Me.SelectSamples.GetCellRange(1, 2, CType(Me.SelectSamples. ✓
Rows.Count, Integer) - 1, 2)
    sampleNamesCellRange.Clip = samples
    SelectSamples.Select(SelectSamples.Row, SelectSamples.Col)

```

```

    Dim i, j As Integer
    j = 1
    For i = 1 To CType(Me.SelectSamples.Rows.Count, Integer) - 1
        If Not SelectSamples(i, 2) Is Nothing Then
            j = j + 1
        End If
    Next

```

```
Next
```

```

    sampleNamesCellRange = Me.SelectSamples.GetCellRange(1, 1, j - 1, 2)
    SampleChoice = sampleNamesCellRange.Clip

```

```
End Sub
```

```
Private Sub btnSelectAll_Click(ByVal sender As System.Object, ByVal e As System. ✓
EventArgs) Handles btnSelectAll.Click
```

```

    Dim i, j As Integer
    j = 1
    For i = 1 To CType(Me.SelectSamples.Rows.Count, Integer) - 1
        If Not SelectSamples(i, 2) Is Nothing Then
            j = j + 1
        End If
    Next

```

```
Next
```

```
    For i = 1 To j - 1
        SelectSamples(i, 1) = True
    Next
```

```
End Sub
```

```
Private Sub btnSelectNone_Click(ByVal sender As System.Object, ByVal e As System.  ✓
EventArgs) Handles btnSelectNone.Click
```

```
    Dim i As Integer
    For i = 1 To CType(Me.SelectSamples.Rows.Count, Integer) - 1
        SelectSamples(i, 1) = False
    Next
```

```
End Sub
```

```
Private Sub btnSelectSamplesOK_Click(ByVal sender As System.Object, ByVal e As System.  ✓
EventArgs) Handles btnSelectSamplesOK.Click
```

```
    Dim sampleNamesCellRange As CellRange
    sampleNamesCellRange = Me.SelectSamples.GetCellRange(1, 2, CType(Me.SelectSamples.  ✓
Rows.Count, Integer) - 1, 2)
    sampleNamesCellRange.Clip = samples
    SelectSamples.Select(SelectSamples.Row, SelectSamples.Col)
    Dim i, j As Integer
    j = 1
    For i = 1 To CType(Me.SelectSamples.Rows.Count, Integer) - 1
        If Not SelectSamples(i, 2) Is Nothing Then
            j = j + 1
        End If
    Next
```

```
Next
```

```
sampleNamesCellRange = Me.SelectSamples.GetCellRange(1, 1, j - 1, 2)
SampleChoice = sampleNamesCellRange.Clip
```

```
End Sub
```

```
End Class
```

Imports Cl.ClPrintDocument

Public Class Text_Output

Inherits System.Windows.Forms.Form

Private myInputText As String

Private myInputVariances As Array

Private myInputT2 As Array

Private myInputpcs As Array

Private myInputnewdata As Array

Private myInputManovap As Array

Private myInputManovad As Array

Private myInputManovastats As Array

Private myInputpdist As Array

Private myInputlinkage As Array

Private myString As String

Public ReadOnly Property StringContents() As String

Get

Return myString

End Get

End Property

Public Property InputText() As String

Get

Return myInputText

End Get

Set(ByVal Value As String)

myInputText = Value

End Set

End Property

Public Property InputVariances() As Array

Get

Return myInputVariances

End Get

Set(ByVal Value As Array)

myInputVariances = Value

End Set

End Property

Public Property InputT2() As Array

Get

Return myInputT2

End Get

Set(ByVal Value As Array)

myInputT2 = Value

End Set

End Property

Public Property Inputpcs() As Array

Get

Return myInputpcs

End Get

Set(ByVal Value As Array)

myInputpcs = Value

End Set

End Property

Public Property Inputnewdata() As Array

Get

Return myInputnewdata

End Get

Set(ByVal Value As Array)

myInputnewdata = Value

```

        End Set
    End Property

    Public Property InputManovap() As Array
        Get
            Return myInputManovap
        End Get
        Set(ByVal Value As Array)
            myInputManovap = Value
        End Set
    End Property

    Public Property InputManovad() As Array
        Get
            Return myInputManovad
        End Get
        Set(ByVal Value As Array)
            myInputManovad = Value
        End Set
    End Property

    Public Property InputManovastats() As Array
        Get
            Return myInputManovastats
        End Get
        Set(ByVal Value As Array)
            myInputManovastats = Value
        End Set
    End Property

    Public Property Inputpdist() As Array
        Get
            Return myInputpdist
        End Get
        Set(ByVal Value As Array)
            myInputpdist = Value
        End Set
    End Property

    Public Property Inputlinkage() As Array
        Get
            Return myInputlinkage
        End Get
        Set(ByVal Value As Array)
            myInputlinkage = Value
        End Set
    End Property

```

#Region " Windows Form Designer generated code "

```

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
    End Sub

```

```

    End If
    MyBase.Dispose(disposing)
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents dataReport As System.Windows.Forms.RichTextBox
Friend WithEvents ContextMenuTextOutput As System.Windows.Forms.ContextMenu
Friend WithEvents ctxCopy As System.Windows.Forms.MenuItem
Friend WithEvents ctxSaveAs As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem
Friend WithEvents ctxPrint As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem5 As System.Windows.Forms.MenuItem
Friend WithEvents ctxExit As System.Windows.Forms.MenuItem
Friend WithEvents TextDoc As Cl.ClPrintDocument.ClPrintDocument
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Dim resources As System.Resources.ResourceManager = New System.Resources.
ResourceManager(GetType(Text_Output))
    Me.dataReport = New System.Windows.Forms.RichTextBox
    Me.ContextMenuTextOutput = New System.Windows.Forms.ContextMenu
    Me.ctxCopy = New System.Windows.Forms.MenuItem
    Me.ctxSaveAs = New System.Windows.Forms.MenuItem
    Me.MenuItem3 = New System.Windows.Forms.MenuItem
    Me.ctxPrint = New System.Windows.Forms.MenuItem
    Me.MenuItem5 = New System.Windows.Forms.MenuItem
    Me.ctxExit = New System.Windows.Forms.MenuItem
    Me.TextDoc = New Cl.ClPrintDocument.ClPrintDocument
    Me.SuspendLayout()
    '
    'dataReport
    '
    Me.dataReport.ContextMenu = Me.ContextMenuTextOutput
    Me.dataReport.Dock = System.Windows.Forms.DockStyle.Fill
    Me.dataReport.Font = New System.Drawing.Font("Times New Roman", 12.0!, System.
Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
    Me.dataReport.Location = New System.Drawing.Point(0, 0)
    Me.dataReport.Name = "dataReport"
    Me.dataReport.Size = New System.Drawing.Size(552, 533)
    Me.dataReport.TabIndex = 0
    Me.dataReport.Text = ""
    '
    'ContextMenuTextOutput
    '
    Me.ContextMenuTextOutput.MenuItems.AddRange(New System.Windows.Forms.MenuItem()
{Me.ctxCopy, Me.ctxSaveAs, Me.MenuItem3, Me.ctxPrint, Me.MenuItem5, Me.ctxExit})
    '
    'ctxCopy
    '
    Me.ctxCopy.Index = 0
    Me.ctxCopy.Text = "&Copy"
    '
    'ctxSaveAs
    '
    Me.ctxSaveAs.Index = 1
    Me.ctxSaveAs.Text = "Save &As"
    '
    'MenuItem3
    '
    Me.MenuItem3.Index = 2
    Me.MenuItem3.Text = "-"
    '
    'ctxPrint
    '

```

```

        Me.ctxPrint.Index = 3
        Me.ctxPrint.Text = "&Print"
        '
        'MenuItem5
        '
        Me.MenuItem5.Index = 4
        Me.MenuItem5.Text = "- "
        '
        'ctxExit
        '
        Me.ctxExit.Index = 5
        Me.ctxExit.Text = "E&xit"
        '
        'TextDoc
        '
        Me.TextDoc.CIDPageSettings = "color:False;landscape:False;margins:100,100,100,100;✓
papersize:850,1100,TABLAHQAdA" & _
        "BLAHIA"
        Me.TextDoc.ColumnSpacingStr = "0.5in"
        Me.TextDoc.ColumnSpacingUnit.DefaultType = True
        Me.TextDoc.ColumnSpacingUnit.UnitValue = "0.5in"
        Me.TextDoc.DefaultUnit = C1.C1PrintDocument.UnitTypeEnum.Inch
        Me.TextDoc.DocumentName = ""
        '
        'Text_Output
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(552, 533)
        Me.ContextMenu = Me.ContextMenuTextOutput
        Me.Controls.Add(Me.dataReport)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.Name = "Text_Output"
        Me.Text = "Text_Output"
        Me.ResumeLayout(False)

```

End Sub

#End Region

```

Private Sub ctxExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles ctxExit.Click
    Me.Close()
End Sub

```

```

Private Sub Text_Output_Click(ByVal sender As Object, ByVal e As System.EventArgs) ✓
Handles MyBase.Click
    Me.Activate()
End Sub

```

```

Private Sub Text_Output_Closed(ByVal sender As Object, ByVal e As System.EventArgs) ✓
Handles MyBase.Closed
    Me.Invalidate()
    Me.Finalize()
End Sub

```

```

Private Sub ctxCopy_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) ✓
Handles ctxCopy.Click
    Dim Selection As String = Me.dataReport.SelectedText
    Clipboard.SetDataObject(Selection)
End Sub

```

```

Private Sub ctxSaveAs_Click(ByVal sender As System.Object, ByVal e As System. ✓
EventArgs) Handles ctxSaveAs.Click
    Dim TextOutput As Text_Output = Me
    Dim saveFileDialog As New SaveFileDialog

```

```

saveFileDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*"
saveFileDialog.FilterIndex = 1
saveFileDialog.FileName = saveFileDialog.FileName
If saveFileDialog.ShowDialog() = DialogResult.OK Then
    TextOutput.dataReport.SaveFile(saveFileDialog.FileName)
End If

```

```
End Sub
```

```

Private Sub dataReport_VisibleChanged(ByVal sender As Object, ByVal e As System.
EventArgs) Handles dataReport.VisibleChanged
    Me.Activate()
End Sub

```

```

Private Sub dataReport_GotFocus(ByVal sender As Object, ByVal e As System.EventArgs)
Handles dataReport.GotFocus
    Me.Activate()
End Sub

```

```

Private Sub ctxPrint_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ctxPrint.Click
    Dim text As String = Me.InputText.ToString
    Dim TextOutput As Text_Output = CType(Me.ActiveMdiChild, Text_Output)
    Dim s As C1.C1PrintDocument.C1DocStyle
    Dim doc As New C1PrintDocument
    With Me.TextDoc
        .Style.Font = New Font("Times New Roman", 12, FontStyle.Regular)
        With .PageHeader
            .RenderText.Style.TextAlignHorz = C1.C1PrintDocument.AlignHorzEnum.Right
            .RenderText.Text = "Header - Page [@@PageNo@@] of [@@PageCount@@]"
            .Height = 0
        End With
        With .PageFooter
            .RenderText.Style.TextAlignHorz = C1.C1PrintDocument.AlignHorzEnum.Right
            .RenderText.Style.TextAlignVert = C1.C1PrintDocument.AlignVertEnum.Bottom
            .RenderText.Text = "Page [@@PageNo@@] of [@@PageCount@@]"
        End With
        .StartDoc()
        .Style.TextColor = Color.Black
        .Style.TextAlignHorz = C1.C1PrintDocument.AlignHorzEnum.Justify
        .RenderBlockText(text)
        .Style.TextAlignHorz = C1.C1PrintDocument.AlignHorzEnum.Left
        .EndDoc()
    End With

    Dim aprev As New Final_Report
    aprev.C1PrintPreview1.Document = Me.TextDoc
    aprev.ShowDialog()
    aprev.Dispose()

```

```
End Sub
```

```
End Class
```

```
Imports Cl.Win.C1FlexGrid
```

```
Public Class Variable_Names
    Inherits System.Windows.Forms.Form
```

```
#Region " Windows Form Designer generated code "
```

```
    Public Sub New()
        MyBase.New()
```

```
        'This call is required by the Windows Form Designer.
        InitializeComponent()
```

```
        'Add any initialization after the InitializeComponent() call
```

```
    End Sub
```

```
    'Form overrides dispose to clean up the component list.
```

```
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
```

```
        If disposing Then
```

```
            If Not (components Is Nothing) Then
                components.Dispose()
```

```
            End If
```

```
        End If
```

```
        MyBase.Dispose(disposing)
```

```
    End Sub
```

```
    'Required by the Windows Form Designer
```

```
    Private components As System.ComponentModel.IContainer
```

```
    'NOTE: The following procedure is required by the Windows Form Designer
```

```
    'It can be modified using the Windows Form Designer.
```

```
    'Do not modify it using the code editor.
```

```
    Friend WithEvents VariableNames As Cl.Win.C1FlexGrid.C1FlexGrid
```

```
    Friend WithEvents btnOpenCompoundList As System.Windows.Forms.Button
```

```
    Friend WithEvents btnSaveCompoundList As System.Windows.Forms.Button
```

```
    Friend WithEvents btnNameVariablesOK As System.Windows.Forms.Button
```

```
    Friend WithEvents Label1 As System.Windows.Forms.Label
```

```
    Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
```

```
    Friend WithEvents btnNameVariablesCancel As System.Windows.Forms.Button
```

```
    Friend WithEvents ContextMenu1 As System.Windows.Forms.ContextMenu
```

```
    Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem2 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem4 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem5 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem6 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem7 As System.Windows.Forms.MenuItem
```

```
    Friend WithEvents MenuItem8 As System.Windows.Forms.MenuItem
```

```
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
```

```
    Dim resources As System.Resources.ResourceManager = New System.Resources.
```

```
    ResourceManager(GetType(Variable_Names))
```

```
    Me.VariableNames = New Cl.Win.C1FlexGrid.C1FlexGrid
```

```
    Me.btnOpenCompoundList = New System.Windows.Forms.Button
```

```
    Me.btnSaveCompoundList = New System.Windows.Forms.Button
```

```
    Me.btnNameVariablesOK = New System.Windows.Forms.Button
```

```
    Me.Label1 = New System.Windows.Forms.Label
```

```
    Me.GroupBox1 = New System.Windows.Forms.GroupBox
```

```
    Me.btnNameVariablesCancel = New System.Windows.Forms.Button
```

```
    Me.ContextMenu1 = New System.Windows.Forms.ContextMenu
```

```
    Me.MenuItem1 = New System.Windows.Forms.MenuItem
```

```
    Me.MenuItem2 = New System.Windows.Forms.MenuItem
```

```
    Me.MenuItem3 = New System.Windows.Forms.MenuItem
```

```
    Me.MenuItem4 = New System.Windows.Forms.MenuItem
```

```
    Me.MenuItem5 = New System.Windows.Forms.MenuItem
```

```
    Me.MenuItem6 = New System.Windows.Forms.MenuItem
```

```
    Me.MenuItem7 = New System.Windows.Forms.MenuItem
```

```

Me.MenuItem8 = New System.Windows.Forms.MenuItem
CType(Me.VariableNames, System.ComponentModel.ISupportInitialize).BeginInit()
Me.SuspendLayout()
'
'VariableNames
'
Me.VariableNames.AllowSorting = C1.Win.C1FlexGrid.AllowSortingEnum.None
Me.VariableNames.BackColor = System.Drawing.SystemColors.Window
Me.VariableNames.ColumnInfo = "1,0,0,0,0,85,Columns:0{Width:172;AllowSorting:False";
Name:"Compound";Caption:"Comp" & _
    "ound";TextAlign:LeftCenter;}" & Microsoft.VisualBasic.ChrW(9)
Me.VariableNames.ForeColor = System.Drawing.SystemColors.WindowText
Me.VariableNames.KeyActionTab = C1.Win.C1FlexGrid.KeyActionEnum.MoveAcross
Me.VariableNames.Location = New System.Drawing.Point(8, 8)
Me.VariableNames.Name = "VariableNames"
Me.VariableNames.Size = New System.Drawing.Size(192, 384)
Me.VariableNames.Styles = New C1.Win.C1FlexGrid.CellStyleCollection("Fixed
{BackColor:Control;ForeColor:ControlText;Border:Flat,1,ControlDark,Both;}" & Microsoft
.VisualBasic.ChrW(9) & "Hi" & _
    "ghlight{BackColor:Highlight;ForeColor:HighlightText;}" & Microsoft.VisualBasic.
ChrW(9) & "Search{BackColor:Highlight" & _
    ";ForeColor:HighlightText;}" & Microsoft.VisualBasic.ChrW(9) & "Frozen{BackColor:
Beige;}" & Microsoft.VisualBasic.ChrW(9) & "EmptyArea{BackColor:AppWorks" & _
    "pace;Border:Flat,1,ControlDarkDark,Both;}" & Microsoft.VisualBasic.ChrW(9) &
"GrandTotal{BackColor:Black;ForeColor:W" & _
    "hite;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal0{BackColor:ControlDarkDark;
ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal1{BackColor" & _
    ":ControlDarkDark;ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal2
{BackColor:ControlDarkDark;ForeColor" & _
    ":White;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal3{BackColor:ControlDarkDark;
ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) & "Subtotal4{BackCol" & _
    "or:ControlDarkDark;ForeColor:White;}" & Microsoft.VisualBasic.ChrW(9) &
"Subtotal5{BackColor:ControlDarkDark;ForeCol" & _
    "or:White;}" & Microsoft.VisualBasic.ChrW(9))
Me.VariableNames.TabIndex = 0
'
'btnOpenCompoundList
'
Me.btnOpenCompoundList.Location = New System.Drawing.Point(240, 200)
Me.btnOpenCompoundList.Name = "btnOpenCompoundList"
Me.btnOpenCompoundList.TabIndex = 1
Me.btnOpenCompoundList.Text = "Open"
'
'btnSaveCompoundList
'
Me.btnSaveCompoundList.Location = New System.Drawing.Point(240, 256)
Me.btnSaveCompoundList.Name = "btnSaveCompoundList"
Me.btnSaveCompoundList.TabIndex = 2
Me.btnSaveCompoundList.Text = "Save"
'
'btnNameVariablesOK
'
Me.btnNameVariablesOK.DialogResult = System.Windows.Forms.DialogResult.OK
Me.btnNameVariablesOK.Location = New System.Drawing.Point(240, 320)
Me.btnNameVariablesOK.Name = "btnNameVariablesOK"
Me.btnNameVariablesOK.TabIndex = 3
Me.btnNameVariablesOK.Text = "OK"
'
'Label1
'
Me.Label1.Font = New System.Drawing.Font("Tahoma", 9.75!, System.Drawing.FontStyle
.Regular, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.Label1.Location = New System.Drawing.Point(228, 24)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(100, 128)
Me.Label1.TabIndex = 4
Me.Label1.Text = "Please type in each compound name you wish to include - in

```

```

elution order"
    '
    'GroupBox1
    '
    Me.GroupBox1.Location = New System.Drawing.Point(216, 168)
    Me.GroupBox1.Name = "GroupBox1"
    Me.GroupBox1.Size = New System.Drawing.Size(128, 128)
    Me.GroupBox1.TabIndex = 5
    Me.GroupBox1.TabStop = False
    Me.GroupBox1.Text = "Compound Lists"
    '
    'btnNameVariablesCancel
    '
    Me.btnNameVariablesCancel.DialogResult = System.Windows.Forms.DialogResult.Cancel
    Me.btnNameVariablesCancel.Location = New System.Drawing.Point(240, 368)
    Me.btnNameVariablesCancel.Name = "btnNameVariablesCancel"
    Me.btnNameVariablesCancel.TabIndex = 6
    Me.btnNameVariablesCancel.Text = "&Cancel"
    '
    'ContextMenu1
    '
    Me.ContextMenu1.MenuItems.AddRange(New System.Windows.Forms.MenuItem() {Me.
MenuItem1, Me.MenuItem2, Me.MenuItem3, Me.MenuItem4, Me.MenuItem5, Me.MenuItem6, Me.
MenuItem7, Me.MenuItem8})
    '
    'MenuItem1
    '
    Me.MenuItem1.Index = 0
    Me.MenuItem1.Text = "Cu&t"
    '
    'MenuItem2
    '
    Me.MenuItem2.Index = 1
    Me.MenuItem2.Text = "&Copy"
    '
    'MenuItem3
    '
    Me.MenuItem3.Index = 2
    Me.MenuItem3.Text = "&Paste"
    '
    'MenuItem4
    '
    Me.MenuItem4.Index = 3
    Me.MenuItem4.Text = "Paste &Special"
    '
    'MenuItem5
    '
    Me.MenuItem5.Index = 4
    Me.MenuItem5.Text = "- "
    '
    'MenuItem6
    '
    Me.MenuItem6.Index = 5
    Me.MenuItem6.Text = "Select &All"
    '
    'MenuItem7
    '
    Me.MenuItem7.Index = 6
    Me.MenuItem7.Text = "- "
    '
    'MenuItem8
    '
    Me.MenuItem8.Index = 7
    Me.MenuItem8.Text = "&Format"
    '
    'Variable_Names
    '

```

```

Me.AcceptButton = Me.btnNameVariablesOK
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
Me.CancelButton = Me.btnNameVariablesCancel
Me.ClientSize = New System.Drawing.Size(352, 413)
Me.ControlBox = False
Me.Controls.Add(Me.btnNameVariablesCancel)
Me.Controls.Add(Me.Label1)
Me.Controls.Add(Me.btnNameVariablesOK)
Me.Controls.Add(Me.btnSaveCompoundList)
Me.Controls.Add(Me.btnOpenCompoundList)
Me.Controls.Add(Me.VariableNames)
Me.Controls.Add(Me.GroupBox1)
Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog
Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
Me.MaximizeBox = False
Me.Name = "Variable_Names"
Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
Me.Text = "Name Variables"
CType(Me.VariableNames, System.ComponentModel.ISupportInitialize).EndInit()
Me.ResumeLayout(False)

```

End Sub

#End Region

```

Private Sub Variable_Names_Load(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MyBase.Load

```

End Sub

```

Public Sub ReturnVariableNames(ByVal VariableNameList As Object)
    Dim mVariableNamesRows As Integer
    mVariableNamesRows = VariableNames.Rows.Count

```

End Sub

```

Private Sub btnOpenCompoundList_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles btnOpenCompoundList.Click
    Dim OpenDlg As New OpenFileDialog
    With OpenDlg
        .FileName = ""
        .Filter = "Variable Name Files (*.vnf)|*.vnf|All files (*.*)|*.*"
        .FilterIndex = 1
        .CheckFileExists = True
        If .ShowDialog() = DialogResult.Cancel Then Return
        VariableNames.LoadGrid(OpenDlg.FileName, FileFormatEnum.TextComma, False)
    End With

```

End Sub

```

Private Sub btnSaveCompoundList_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles btnSaveCompoundList.Click
    Dim SaveAsDlg As New SaveFileDialog
    With SaveAsDlg
        .FileName = ""
        .Filter = "Variable Name Files (*.vnf)|*.vnf|All files (*.*)|*.*"
        .FilterIndex = 1
        If .ShowDialog() = DialogResult.Cancel Then Return
        VariableNames.SaveGrid(SaveAsDlg.FileName, FileFormatEnum.TextComma, False)
    End With

```

End Sub

```

Private Sub VariableNames_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles VariableNames.Click

```

End Sub

```
Private Sub btnNameVariablesOK_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles btnNameVariablesOK.Click
End Sub
```

```
Private Sub VariableNames_KeyDown(ByVal sender As Object, ByVal e As KeyEventArgs)
Handles VariableNames.KeyDown
```

```
    Dim copy As Boolean, paste As Boolean, cut As Boolean
    ' ** copy: ctrl-C, ctrl-X, ctrl-ins
    If e.Control Then
        If e.KeyCode = Keys.C Or _
            e.KeyCode = Keys.Insert Then
            copy = True
        End If
        If e.KeyCode = Keys.X Then
            cut = True
        End If
    End If
    ' ** paste: ctrl-V, shift-ins
    If (e.Control = True And e.KeyCode = Keys.V) Or _
        (e.Shift And e.KeyCode = Keys.Insert) Then
        paste = True
    End If
    ' ** copy selection to clipboard
    If copy Then
        Clipboard.SetDataObject(VariableNames.Clip)
    End If
    ' ** cut selection from clipboard
    If cut Then
        Clipboard.SetDataObject(VariableNames.Clip)
        Dim selected As Cl.Win.ClFlexGrid.CellRange
        selected = VariableNames.Selection
        selected.Data = Nothing
    End If
    ' ** paste from clipboard
    If paste Then
        ' see if there's text in the clipboard
        Dim data As IDataObject = Clipboard.GetDataObject()
        If data.GetDataPresent(DataFormats.Text) Then
            ' there is, so paste it
            VariableNames.Select(VariableNames.Row, VariableNames.Col, VariableNames.
Rows.Count - 1, VariableNames.Cols.Count - 1, False)
            VariableNames.Clip = CType(data.GetData(DataFormats.Text), String)
            VariableNames.Select(VariableNames.Row, VariableNames.Col)
        End If
    End If

    'If the user presses the delete key in a cell or in a range of cells, delete them
    If e.KeyCode = Keys.Delete Then
        Dim selected As Cl.Win.ClFlexGrid.CellRange
        selected = VariableNames.Selection
        selected.Data = Nothing
    End If
End Sub
```

```
End Class
```