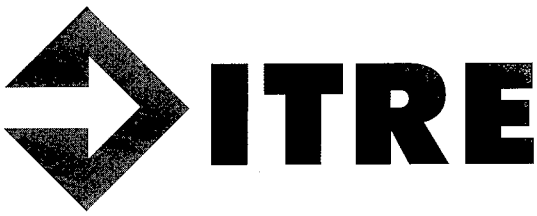


Prototype Demonstration of a Geographic Information System Application for the Seasonal Analysis of Traffic Data, Development of Seasonal Factors and Seasonal Adjustment of Roadways

Final Report

Dr. Gorman Gilbert
director

A joint activity
of North Carolina
universities

NC A&T State University

NC Central University

NC State University

UNC Chapel Hill

UNC Charlotte

Duke University

Institute for Transportation Research and Education
North Carolina State University
Centennial Campus
Box 8601, Raleigh, NC 27695-8601
(919) **515-8899** (919) 515-8898 fax
http://itre.ncsu.edu/

ITRE
Institute for Transportation Research and Education
North Carolina State University

Prototype Demonstration of a Geographic Information System Application for the Seasonal Analysis of Traffic Data, Development of Seasonal Factors and Seasonal Adjustment of Roadways

Final Report

Dr. Gorman Gilbert
director

A joint activity
of North Carolina
universities

NC A&T State University
NC Central University
NC State University
UNC Chapel Hill
UNC Charlotte
Duke University

Prepared By:

Shannon M. McDonald, PI

of the

Geographic Information Systems Program
Institute for Transportation Research and Education
Raleigh, NC

15 August 1999

Centennial Campus
Box 8601, Raleigh, NC 27695-8601
(919) **515-8899** ☎ (919) 515-8898 fax
http://itre.ncsu.edu/

Technical Report Documentation Page

1. Report No. FHWA/NC/99-008	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Prototype Demonstration of a Geographic Information System Application for the Seasonal Analysis of Traffic Data, Develpment of Seasonal Factors and Seasonal Adjustment of Roadways		5. Report Date August 1999 6. Performing Organization Code
7. Author(s) Shannon M. McDonald, Principal Investigator		8. Performing Organization Report No.
9. Performing Organization Name and Address Institute for Transportation Research and Education Centennial Campus, Box 8601 Raleigh, NC 27695-8601		10. Work Unit No. (TRAIS) 11. Contract or Grant No.
12. Sponsoring Agency Name and Address North Carolina Department of Transportation P.O. Box 25201 Raleigh, NC 27611		13. Type of Report and Period Covered Final Report July 1, 1998 to June 30, 1999 14. Sponsoring Agency Code HWY 99-5

15. Supplementary Notes

16. Abstract

The Traffic Survey Unit plans to establish a methodology in which it can assign each Portable Traffic Counter (PTC) station a seasonal group profile through a means of statistical and geographical analysis. An ArcView Geographic Information System application was programmed to allow the traffic survey sites to be analyzed by their spatial nature as well as by their statistical profiles. This program was created using the Avenue programming language.

The program offers a user-friendly GIS interface that can aid analysts with updating the current seasonal groupings for traffic survey points across North Carolina. The user is able to select a site and display an attribute table listing its current seasonal assignment, its new seasonal assignment based on the best "best guess" as determined by the statistical analysis, and all the "best guesses" determined by that analysis. The user is also able to summon displays of the seven Seasonal Profiles, and the graphs and charts produced by the statistical analysis. Finally, the user is able to update the new seasonal assignment based on the results of the statistical analysis and its geographical proximity to the other stations, highways, and municipalities.

17. Key Words GIS, Seasonal, Traffic Survey Highway, PTC, ATR, ITRE	18. Distribution Statement No restrictions. 150 copies of this document were printed at cost of $4.83 per copy.		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 158	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

The contents of this report reflect the views of the Author and not necessarily the views of the University. The Author is responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the North Carolina Department of Transportation nor the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

TABLE OF CONTENTS

GLOSSARY OF TERMS

ArcView project file (.apr)

An ASCII formatted file that sets the parameters for an ArcView session. If a new project is opened, ArcView's default .apr file is read, opening a blank project with the system default variables. When a project is customized and saved, a new project file is created. This file is read during the startup of the project. The file contains, among other things, a set of pointers to the project components (documents).

attribute (item, field)

A character of a feature, usually stored as values in columns in a table. Items, fields, and attributes are synonymous terms.

Automatic Traffic Recorder (ATR) Station

Permanent traffic recorder for the continuous collection of traffic data.

avenue

The programming language and customization and development environment of ArcView. Avenue is commonly used to write program scripts to automate repetitive tasks, modify and customize the ArcView interface, develop and distribute new applications, and integrate ArcView with other applications. Avenue is a hybrid language that has characteristics of both procedural and object oriented languages.

best guess group assignments ("best guess")

The seasonal profile curve that best correlates with the data collected at a Portable Traffic Counter site.

coverage (cover)

The data structure of geographic features as stored by ARC/INFO. A coverage may contain one or more primary features such as points, lines, and polygons, and secondary features such as tics, links, or annotation. These features are usually associated with records in a feature attribute table. A typical coverage might be

lines (arcs) that represent the individual segments of a road network, with an attribute table containing items describing the route number, number of lanes, street name, etc.

feature

A geographic representation of an entity on Earth. A feature can be an arc (line), polygon, or a point. Secondary features include, among other things, tics, annotation, and links. Features are usually associated with one record in a feature attribute table.

feature (theme) attribute table

Database table comprised of attributes (columns) and records (rows). Each record in a feature attribute table represents a geographic entity (feature) in a theme. The intersection of a record and attribute is the value of that particular attribute for the feature the row represents.

hot link

Link between a file and the features in a theme. A hot link gives the user the ability to associate each feature in a theme with some action, such as opening an image window with a linked image file displayed.

Portable Traffic Counter (PTC) Station

Location where a short-term count is taken for the purposes for publishing an average annual daily traffic (AADT) count.

shapefile

The data structure of geographic features as stored by ArcView. A shapefile may contain one or more features such as points, lines, and polygons. These features are usually associated with records in a feature attribute table.

theme

A collection of features drawn in a view represented by symbols in a classifiable legend. The features in a theme have a common relationship. For example, the features in a theme may be a collection of points that all represent fire hydrants.

The list of themes available in a View GUI are listed in the view's table of contents, an area left of the digital map area. Sources of themes include shapefiles, coverages, and images. Computer Aided Drafting (CAD) files can also be represented as a theme in a view.

value

The intersection of a record and an item (attribute) in a table. The value is an answer to the attribute for the record in question. For example, a selected records value for the Median Household Income attribute may be $100,000.

Executive Summary

Traffic engineers, transportation planners, and countless other professionals need accurate, timely data to make proper decisions. Whether one is considering re-timing a signal, widening an arterial, or building a new freeway, current traffic volumes are among the most critical pieces of information required. The Statewide Planning Branch of the Division of Highways of the North Carolina Department of Transportation (NCDOT) collects traffic volumes at thousands of locations along North Carolina's roadways.

The project's primary objectives are to benefit the:

1. Update and validation of existing seasonal roadway assignments used in the estimation of annual average daily traffic volumes,
2. Update and validation of seasonal traffic adjustment factors and the seasonal grouping of continuous data collections sites,
3. Accuracy and validity of traffic demand models and traffic forecasts, and
4. Accuracy and validity of all traffic data being provided to the NCDOT, urban transportation agencies, and other transportation management systems.

The first primary task was the reassignment of the current seasonal groupings to each of the approximately 200 short-term counting stations along the US Highway 70 corridor in eastern North Carolina. Dr. Tom Johnston of the Department of Statistics at North Carolina State University examined the methodology and data requirements of the Statewide Planning Branch and developed recommendations to the preferred data collection and statistical analysis procedure for the project.

An ArcView Geographic Information System application was programmed to allow the traffic survey sites to be analyzed by their spatial nature as well as by their statistical profiles. This program was created using the Avenue programming language.

The program offers a user-friendly GIS interface that can aid analysts with updating the current seasonal groupings for traffic survey points across North Carolina. The user is able to select a site and display an attribute table listing its current seasonal assignment, its new seasonal assignment based on the best "best guess" as determined by the statistical analysis, and all the "best guesses" determined by that analysis. The user is also able to summon displays of the seven Seasonal Profiles, and the graphs and charts produced by the statistical analysis. Finally, the user is able to update the new seasonal assignment based on the results of the statistical analysis and its geographical proximity to the other stations, highways, and municipalities.

1.0 INTRODUCTION

One of the most significant functions of highway and planning management is the measurement of traffic volumes. Today we are experiencing a period of consistent traffic growth across North Carolina. With this increase in traffic flow, it is necessary to constantly improve and update transportation management systems. To this end, the North Carolina Department of Transportation (NCDOT) Traffic Survey Unit wishes to update its seasonal grouping assignments for over 58,000 traffic count stations throughout the state.

The updated seasonal grouping assignments are needed to prepare future-year traffic forecasts that are more credible and more useful in preparing financially feasible transportation plans. These traffic counts not only provide a method to accurately count annual average daily traffic, but they can also be used in other traffic analyses. Examples include peak hour estimates, traffic flow, capacity analysis, traffic engineering analysis, street and intersection design, and improved safety systems.

1.1 PROBLEM STATEMENT

The Division of Highways is required to conduct seasonal analysis of traffic volume data, vehicle classification, and vehicle weight data, a requirement established by the Intermodal Surface Transportation Efficiency Act (ISTEA) '91 and the Federal Highway Administration (FHWA) Traffic Monitoring Guide. The current seasonal grouping assignments of the traffic collection points needed to be updated to ensure accurate analysis. The current group assignments to the traffic count station locations were twenty years out of date.

The Traffic Survey Unit plans to establish a methodology in which it can assign each Portable Traffic Counter (PTC) station a seasonal group profile through a means of statistical and geographical analysis.

1.2 PROJECT GOALS

There were to be two major significant outcomes of this proposed work. The first significant outcome was to document a statistical method to estimate the correct seasonal profile of a roadway segment. The second outcome was to obtain a personal computer-based geographic information system application that will reduce the staff time required to check, validate and maintain seasonal group assignments for fifty-eight thousand PTC sites.

1.3 RESEARCH METHODOLOGY AND TASKS

I. Develop a method by which readings of 7-14 days each should be taken at any average count site.

a. The method will identify the number of days per reading necessary to produce a statistically valid sample that will determine the best fit of the data to one of the current seasonal profiles. We have enlisted the services of an NCSU statistician, Dr. Tom Johnson, to advise this phase of the project.

b. A documented technical and literature research will be conducted to find the best method, software, and process to follow for this and subsequent tasks.

c. Create a report detailing the proposed method, and assist NCDOT in its implementation.

d. Deliverable for this task: Technical Report

II. Develop an application to assign a "best guess" seasonal group to each site in the pilot area, based on the most recent data at that site.

a. The "guess" method must consider the best fit of the periodic readings against the current snapshot of the current seasonal groupings.

b. The application will be designed to allow the use of new seasonal profiles when they are updated by NCDOT's Traffic Survey Unit.

c. The application will be run when a global reassignment of seasonal groupings is desired, or to calculate smaller groups of reassignments when new count data arrives from the field. Refinement of the assignments of individual stations will be accomplished with a separate application described in task #3.

d. Work will begin with the data from the 100 ATR sites, whose seasonal group assignment is known, as control data to refine this application. ITRE has enlisted the services of Dr. Tom Johnson to advise this phase of the project. His guidance will be necessary to develop the proper algorithm for fitting the station data to the seasonal profiles.

e. The application shall be designed to operate on Intel-based hardware under Microsoft Windows NT 4.0 operating system.

f. Deliverables for this task:

1. Seasonal grouping assignment application

2. Application documentation

3. Data set delineating seasonal assignment grouping for each input traffic count station, in a format appropriate for supplying input to the application described in task #3.

III. Develop an ArcView application to display roads, the 100 Automatic Traffic Recorder (ATR) sites, and the traffic count sites classified by the new "best guess" grouping values. Many ArcView tools, such as those for zooming, panning, selecting, etc., will be incorporated.

a. A provision will be made to display the current grouping (i.e., pre-"best guess" grouping).

b. The program will also allow the incorporation of additional sites as new counts come in from the field, and are assigned seasonal groupings by the application described in task #2.

c. The application will be used by a DOT analyst to change the assigned values based on the proximity of adjacent stations. The analyst will also be able to visually compare a displayed graph of the periodic collections at that point against the profile of the "best guess" (by default) and the profiles of the other six seasonal groupings (selectable).

d. The user will also be able to examine the seasonal profile of the ATR(s) located in the county being examined and its seasonal history over a five-year period. The user will be able to view the data by county, by Region, by seasonal grouping, or by roadway.

e. The application shall be flexible to allow both the increase and decrease of count stations during the year, as well as allow either the increase or decrease of seasonal groups on an annual basis.

f. The application will be developed in conjunction with NCDOT Traffic Survey Unit personnel. Interface testing will be conducted, if desired, to ensure maximum functionality and flexibility in terms of its operation and use.

g. The application shall be designed to operate on Intel-based hardware under Microsoft Windows 95/98. If not practical, alternate workstation configurations shall be recommended (however, this eventuality is not anticipated).

h. ITRE will provide documentation for this application and train NCDOT personnel in the use of the application.

i. Deliverables for this task:

1. Interactive ArcView application

2. Application documentation

3. Application training

2.0 PROJECT OVERVIEW

2.1 OVERVIEW OF THE ROLES AND RECOMMENDATIONS OF THE DEPARTMENT OF STATISTICS

Dr. Tom Johnson, of the Department of Statistics at North Carolina State University, has examined the current counting methodology and data requirements of the Statewide Planning Branch. Through his analysis, he has developed recommendations as to the preferred data collection and analysis procedure for this project.

Dr. Johnson focused on the first proposal task: The development of a method by which traffic count readings should be taken at any short-term count site. Dr. Johnson researched the methods used within NCDOT to take short-term readings, the methods used by other states that responded to an e-mail query, and studied the short-term count data from 1991. He concluded that there could be no absolute number of readings at a short-term site that can be guaranteed to yield a sufficient sample by which to assign it a seasonal grouping. He recommends a baseline set of four to six readings be taken one year, and that data should be compared against previous years' readings to ascertain each site's variability. The next year's reading should then be planned based on each site's variability from the predicted volume based on the best seasonal group assignment, with sites exhibiting high variability requiring more than the baseline number of counts. A detailed report of Dr. Johnson's recommended activities to be taken in assigning groups is attached to this report as *Attachment A*.

Dr. Johnson also focused on the second proposed task: The development of an application to assign a "best guess" seasonal group to each site in the pilot area, based on the most recent data at that site.

The project proposal states that an application to assign a "best guess" seasonal group to each short-term site based on recent traffic volumes would be created. Originally, this process was intended to be an automated, or non-interactive batch mode process. However, a process that involved an interactive data-exploring interface was recommended and utilized. This framework supports informed engineering judgement for group assignment, and makes it easier for the analyst to investigate the integrity of the data. SAS's JMP Statistical Discovery software was selected by Dr. Johnson as the most appropriate statistical software to use for the analysis of the traffic count data. The text of the justification statement is attached to this report as *Attachment B*. The response to this report was only that a clarification be made of a minor point, so Dr. Johnson's recommendation of JMP was adopted as the platform of choice for the analysis.

The decision to recommend JMP was based on the software package's flexibility in allowing visualization of the data in multiple ways as well as its affordability. Other packages, while easier to program in batch mode, were reviewed and rejected because of their lack of flexibility and expense. One requirement of JMP is that assignments of a

"best guess" to each traffic count station must be done using "cookbook" point-and-click interactive analysis. Dr. Johnson developed such a procedure, detailing the steps necessary to produce the "best guesses" with supporting charts and tables for each station, using Pairwise Correlation analysis in JMP (*Attachment C*).

The results from the JMP analysis were stored as GIF format images. These images are retrievable in ArcView by an established hot link between the PTC site features and the files (see section 3.2 for the hot link procedures). Several trials were conducted for the process of capturing the output information, though none proved significantly more efficient over the other. A final approach to the capture was to use a template created in Unisys Corporation's Paint Shop Pro 4.1. The tutorial for capturing these images (*Supplement B*) uses a similar approach with Microsoft's Photo Shop.

2.2 OVERVIEW OF THE ROLE OF THE INSTITUTE FOR TRANSPORTATION RESEARCH AND EDUCATION

For the purposes of this project, ITRE began with the current seven seasonal grouping profiles. Our primary task then became the reassignment of the current seasonal groupings to each of the approximately 200 short-term counting stations along the US Highway 70 corridor in eastern North Carolina. Dr. Tom Johnston of the Department of Statistics at North Carolina State University examined the methodology and data requirements of the Statewide Planning Branch and developed recommendations to the preferred data collection and analysis procedure for the project.

Although we recommended the *best* seasonal grouping (using statistical correlation and their associated p-values), a group profile was also recorded if it fell within a 5% probability of the best grouping.

The third task of the proposal was under the purview of the Institute. ITRE developed an ArcView application to display roads, the 100 ATR sites, and the traffic count sites with the "best guess" grouping value displayed. The application was built in its 'beta' form to display the Statewide Primary Roads network, the 100 ATR count sites, railroads, water features, and NC County boundaries. The locations of the short-term count stations in the pilot corridor are also displayed. The user is able to select a PTC site and display an attribute table listing its current seasonal assignment, its new seasonal assignment based on the best "best guess" as determined by the analysis in Task 2, and all the "best guesses" determined by that analysis. The user is also able to display the seven Seasonal Profiles, and the graphs and charts produced by the statistical analysis in Task 2. Finally, the user is able to update the new seasonal assignment based on the results of the statistical analysis and its spatial proximity to other stations.

Another product of the program was the training of the project staff of both the basic concepts and applications of ArcView and the utilization of the beta version of the application. The former is a routine class taught by the GIS staff at ITRE. The following description of the two-day class is an excerpt from the training brochure for the GIS/GPS Program at ITRE:

Visualizing Data with ArcView
This two-day course teaches students how to use ArcView, the menu-driven geographic visualization and query program for the desktop. Students will learn to access and update geographic data sets, create and manipulate themes, geocode addresses, design map layouts, and generate hard copy maps. The course is designed for students who primarily wish to work with existing data sets as opposed to developing their own. In addition, the user-friendly interface is well suited for first time GIS users.

A second one-day seminar was held to train the project staff and analysts with the customized ArcView interface and the SAS JMP software. The two tutorials are supplemental to this report (*Supplement A* and *B*, respectively).

2.3 CREATING THE INTERACTIVE ARCVIEW APPLICATION

An alpha version of the ArcView application was developed during the fourth quarter of 1998. The application contained a view component with themes and associated tables depicting highways, county boundaries, railroads, surface water features, municipalities, labeled ATR sites, an interactive locator map, and several Avenue scripts associated with a customized Graphical User Interface (GUI). The documentation for using the interface was delivered as an attachment to the quarterly report on December 31, 1998 (*Attachment D*). Prior to the report, a meeting was held to present the core project group with the alpha version of the application and to solicit the group's feedback. The design notes suggested by the group were implemented to create the first beta version of the application (*Attachment E*). This was followed by a series of meetings and application demonstrations, each of which was followed by a list of change requests that were added to the program when possible.

The DOT wanted Microsoft Access® to be at the center of the application (not a .dbf table) since the ATR and PTC data was already stored in that format. However, unlike .dbf files, tables in Access can not be directly edited in ArcView. Thus, the primary focus of the program was to implement a procedure that allowed the analyst to select and edit records stored in an Access format (.mdb) file through an ArcView interface. The idea was to match the PTC site location features stored in an ArcView readable shapefile (complete for those sites along the US Highway 70 corridor) with the Access table (PTC inventory table) that contained the attribute information for those sites. This would allow the sites to be analyzed by their spatial nature and selected using the graphical selection tools supplied by the ArcView program. Once selected, edits would then need to be made to the table via a customized dialog box and a series of SQL commands. By joining the PTC inventory table to the PTC attribute table, we were able to classify the sites by pending group assignment and acceptance value, and by whether or not data was available for the site. This procedure is described in more detail in the section about the Classify field (section 3.1.1).

Other functions of the program called for an *on-the-fly* hot link between the PTC site features and their associated SAS JMP output images. In addition, the user needed to view the seven seasonal group profile images. The PTC stations needed to be classified by their pending PGA status and whether or not they had been "accepted" by the analyst. Likewise, the ATR stations needed to be classified by their current seasonal profile group assignment. During the latter part of project development, a request to annotate the PTC stations was made, in addition to annotating the ATR stations.

Finally, careful thought had to go into guarding the program from user-error. Saving and other procedures that allow a user to manipulate the program interface and/or scripts are guarded by a password message box. For example, anyone who wishes to open the Customize dialog box or save the project must have the appropriate password. The database is also safeguarded from a user's attempt to access the PTC inventory database after the reset date through the ArcView program. A warning dialog displays during project startup once within thirty days of the reset date. Once expired, access to the program is only given to those who obtain the password.

Once all design recommendations were implemented, the final step was to upload the program onto the DOT Traffic Survey Unit's server. Once in place the shapefiles that were used for geographic landmarks (highways, railroads, surface waters, etc.) were deleted and replaced with those on their server. A step-by-step guide attached as *Attachment F* illustrates how the program was loaded onto the department's Windows NT-platform personal computer.

The following section describes the program functionality in more detail.

3.0 THE CUSTOMIZED ARCVIEW INTERFACE

Several versions of this program have undergone review by the program staff. Key personnel evaluated each version and comments and requests for change were noted. When possible, the requests were implemented resulting in a user-friendly GIS interface whose goal is to ultimately aid in the update and validation of existing seasonal roadway assignments and improve the accuracy of all traffic data being provided to the NCDOT, urban transportation agencies, and other transportation management systems.

3.1 THE FLOW OF DATA BETWEEN MS-ACCESS AND THE ARCVIEW INTERFACE

Recall that the program is focused around an MS-Access database table that contains attribute information about the PTC sites. The table is imported into the project via an Open DataBase Connectivity (ODBC) link between ArcView and MS-Access. To establish this connection, an ODBC driver (Microsoft Access Driver version 3.50.360200) was installed and named "seascov" using Microsoft's ODBC Administrator. Through ArcView's supplied SQL connection dialog box (Figure 3.1), records inside the table are queried and opened as *Table1*. The connection only needs to be established

during the initial project upload. If the table does not exist at the project opening, a warning message is displayed and a password request is prompted.

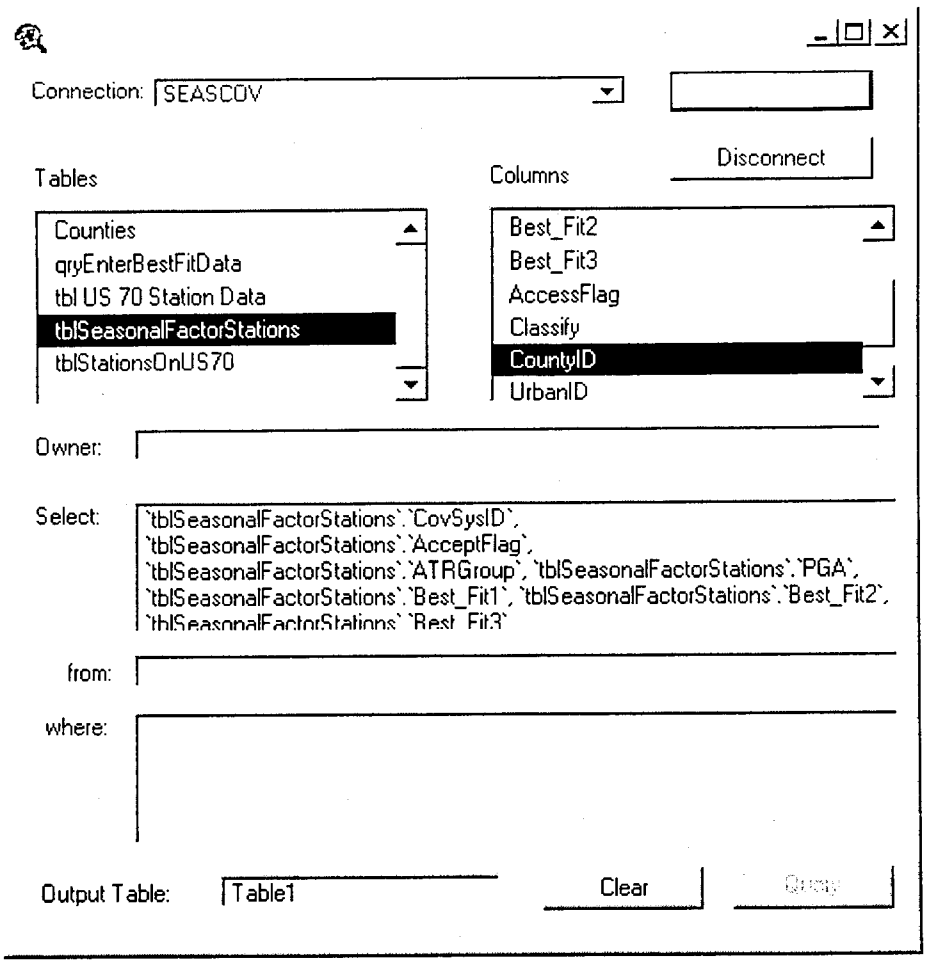


Figure 3.1: ArcView's SQL Dialog Box

Note that the database's logical and physical location is irrelevant for the purposes of the SQL connection. The pathname that holds the database is established in the ODBC configuration accessed through Microsoft's ODBC Administrator.

3.1.1 THE ACCESS PTC INVENTORY DATABASE TABLE

The items queried from the PTC inventory table become the attributes of *Table1*. Most have essential roles in the ArcView program; others are for reference. The following paragraphs describe each item and its role or roles in the program.

CovSysID. The values of this integer-type attribute are unique for each PTC site. The unique identifier establishes a means of selecting and joining records from and to *Table1*. It is crucial that the values for this attribute always remains unique in order for the program to work correctly.

The CovSysID field was created by the Traffic Survey Unit during the program creation. Prior to the creation of CovSysID, sites were identified by their CountyID and StationID fields, neither of which is unique by itself (StationID is not queried for *Table1*).

AcceptFlag. This integer-type attribute indicates that the user has or has not accepted the PGA value for a given record. The number will either be a zero (0) or an eight-digit number representing the date of acceptance in the format YYYYMMDD. When the value for a record's AcceptFlag field is zero, the PTC site the record represents has not been accepted.

ATRGroup. The ATRGroup field is a single binary byte integer that indicates the "current" seasonal profile group assignment. This historical information is included in the table as a source of information to aid in the analysts decision for updating the pending group assignment. Initially the values for the PGA field are calculated from the ATRGroup field, and changed only when new data exists for the site and the "best guess" profile differs from the ATRGroup's value.

PGA. The Pending Group Assignment (PGA) field values represent the seasonal profile group updates that are to be assigned to the PTC sites' ATRGroup during the yearly reset.

This field is the primary focus for the editing process. Initially, all site PGA values are the same as those of the ATRGroup assignment. If data is present for the site, then the SAS analysis is run and a "best guess" profile is found. Those sites whose best guess is different from their ATRGroup assignment will acquire that group number as their PGA. The analyst then has the option to change this assignment to any of the seven profiles. During the annual reset, the PGA assignments are used to calculate the new ATRGroup values. In turn, the outdated ATRGroup values are kept in the PTC inventory as historical fields.

Best_Fit1. The single-byte binary integer in this field indicates the seasonal profile for which the site's data best fits. This information is obtained during the SAS JMP analysis and is only present (greater than zero) when a site has new data that correlates within five percent with at least one of the seasonal profiles. When present this "best guess" assignment is calculated for the record's PGA.

Best_Fit2. Similar to Best_Fit1, the integer indicates the second-best correlation between the site's data and the seasonal profiles. The value is only greater than zero if more than one profile correlates within five percent.

Best_Fit3. The values for Best_Fit3 are similar to Best_Fit1 and Best_Fit2, but only occur when three or more profiles correlate to the site data within five percent. As with Best_Fit2, the data in Best_Fit3 is used only for aiding the analyst in his or her decision.

AccessFlag. The AccessFlag is a character-type attribute whose values are boolean in nature (either Y or N). When the AccessFlag is "raised", that is when the value is "Y",

the site that the record represents obtains new data. When this occurs, the PGA value is the same as the best guess value in Best_Fit1.

Classify. This field was added during the latter part of the program creation. The values of Classify dictate how the PTC sites appear in the SGAA (Seasonal Group Assignment Application) View. Prior to the incorporation of this field, two PTC themes were needed in the View; one classified by the PGA to showing the current group assignment, and one by the AcceptFlag showing whether or not the station had been accepted. The latter had its features viewable as crosshairs whenever the AcceptFlag was greater than zero (whenever a date was present). When overlaid on top of the PGA-classified theme, the analyst would be able to note the stations current PGA assignment and acceptance status.

The problem with this method is that it required joining both PTC attribute tables to the imported PTC inventory table. This procedure took place each time the tables were refreshed, a process unavoidable in order to display the features in the SGAA View with their appropriate values. Part of the solution to this problem was to create a new field, Classify, that allowed the PTC sites to be classified into 5 possible categories:

1. Sites that have new data but have not been accepted,
2. Sites that have new data and have been accepted,
3. Sites that have no new data and have not been accepted,
4. Sites that have no new data but have been accepted, and
5. Sites that are not assigned.

The Classify field is calculated for each record in the Update PGA dialog box each time edits are applied. In all, there are 29 unique possibilities for Classify; 7 for each of the first four categories and one for the last. The calculations are set as a series of *if...then* statements in Avenue that run when edits are applied to the PTC inventory table. The following is an excerpt from that script (refer to script documentation in *Supplement C: Script Documentation*):

From MyDialog.Lbt_Apply.Click

Where `anAccessVal` is the value for AccessFlag,
`anAcceptVal` is the value for AcceptFlag,
`aPGAValNum` is the value for PGA,

```
if (anAccessVal = "0") then
  if (anAcceptVal = "0") then
    aNegPGAVal = (aPGAValNum * -1).AsString
    UpdateClass = "UPDATE tblSeasonalFactorStations SET
                  tblSeasonalFactorStations.Classify ="
                  ++aNegPGAVal++ "WHERE" ++anExpression++ ";"
    theSQLCon.ExecuteSQL(UpdateClass)
  else
    aNegPGAVal = (aPGAValNum * -100).AsString
    UpdateClass = "UPDATE tblSeasonalFactorStations SET
                  tblSeasonalFactorStations.Classify ="
                  ++aNegPGAVal++ "WHERE" ++anExpression++ ";"
```

```
        theSQLCon.ExecuteSQL(UpdateClass)
      end
    else
      if (anAcceptVal = "0") then
        UpdateClass = "UPDATE tblSeasonalFactorStations SET
                      tblSeasonalFactorStations.Classify ="
                      ++aPGAVal++ "WHERE" ++anExpression++ ";"
        theSQLCon.ExecuteSQL(UpdateClass)
      else
        aPGAVal = (aPGAValNum * 100).AsString
        UpdateClass = "UPDATE tblSeasonalFactorStations SET
                      tblSeasonalFactorStations.Classify ="
                      ++aPGAVal++ "WHERE" ++anExpression++ ";"
        theSQLCon.ExecuteSQL(UpdateClass)
      end
    end
```

The resulting 29 unique values that the Classify field can obtain are:

1 thru 7 by 1	(accessflag is 1, acceptflag is 0)
-1 thru -7 by 1	(accessflag is 0, acceptflag is 0)
100 thru 700 by 100	(accessflag is > 0, acceptflag is 1)
-100 thru -700 by 100	(accessflag is > 0, acceptflag is 0)

CountyID. The values indicate to which county each site is affiliated. The values of this field were used in conjunction with those of the StationID to create values for CovSysID. This field is also used during the initial selection of *Table 1*'s records by county during the project startup. The next section describes this process in more detail.

3.1.2 SELECTING PTC SITES BY COUNTY: CREATING *QTABLE*

The first beta version of the program joined *Table1* to two PTC attribute tables. Recall that *Table1* table was imported into ArcView, and both the attribute tables and *Table1* are large (around 58 thousand records each). When executed, a refresh on the PTC theme or attribute table queries and re-imports *Table1* and all associated joins are reestablished. This made the editing process inefficient, taking up to fifteen minutes to execute.

The first step to reducing the editing time was to reduce the number of joins. Deleting one of the attribute tables by adding a new field, Classify, to the PTC inventory did this (see section 3.1.1). Now only one join was needed, cutting the time almost in half.

The second step proved more significant. By prompting the user to select which county or counties with whom he or she needed to work, *Table1* could be queried to create a much smaller query table. This could be done because the area that an analyst observes is usually smaller than a single county. The smaller table, named *qtable*, contains the records for each county selected by the user, and all of the attributes and values for those records. The table is joined to the PTC theme attribute table, and during a refresh, only this VTab is rejoined. Furthermore, no SQL connection has to be reestablished. This

again reduces the editing time, this time by factor of over 400, taking less than one second per record to execute.

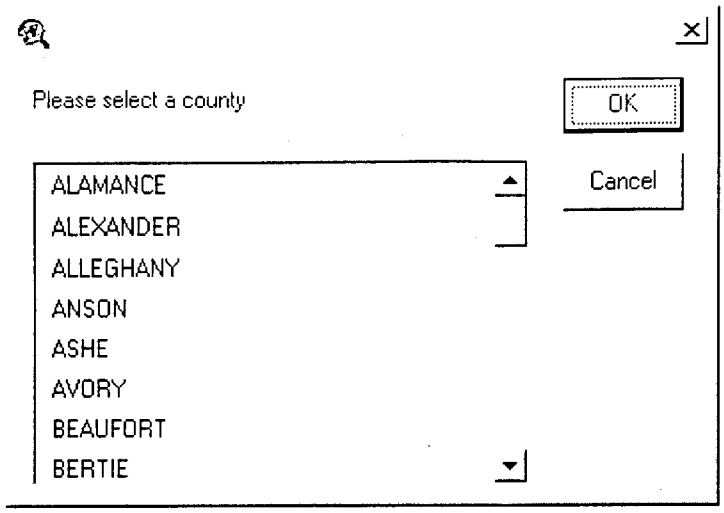


Figure 3.2: County Selection Query Dialog Box

Since *Table1* is not refreshed during the editing process, the values of its records do not reflect changes that have taken place. However, *Table1* is only needed to query *qtable*, so it is only important that the values be correct at project startup.

3.1.3 EDITING THE PTC INVENTORY TABLE: THE PGA UPDATE DIALOG BOX

The PGA Update dialog box is the vehicle by which records in the PTC inventory table in MS-Access can be viewed and edited from the ArcView platform.

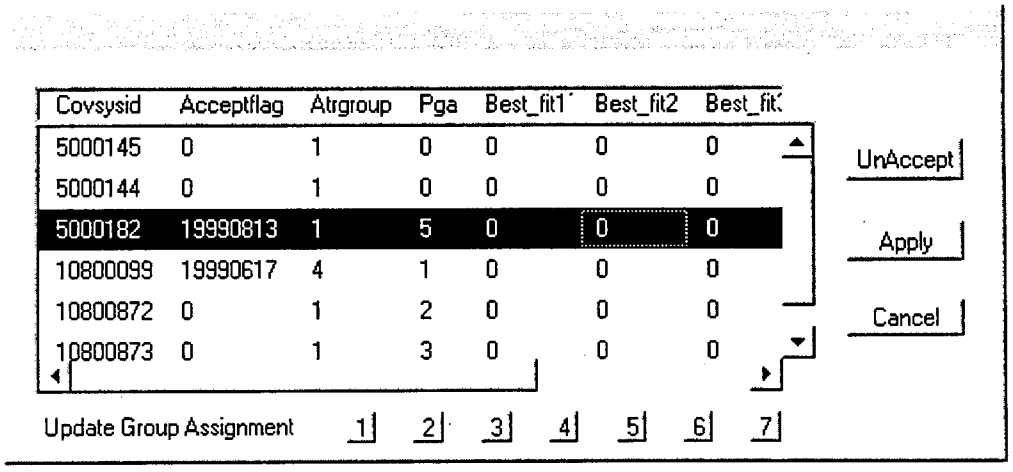


Figure 3.3: PGA Update Dialog Box

The dialog box was created using ArcView's Dialog Designer extension, an interface that allows the creation of controls and the association of their properties with scripts, text, or files. Examples of controls are buttons, check boxes, list boxes, and control panels. The PGA Update dialog box is composed of eleven label buttons, one list box, one text label, and the dialog.

Dialog control settings for the PGA Update dialog box:

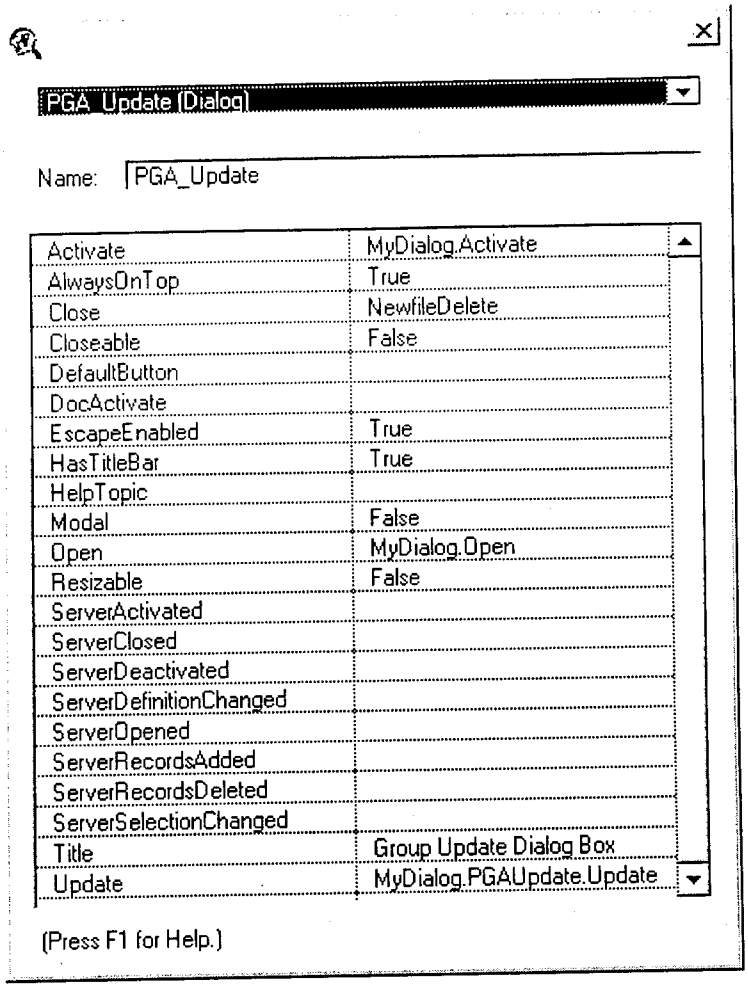


Figure 3.4: PGA Update dialog box dialog control properties

- AlwaysOnTop -- when TRUE, the dialog box always promotes to the top of the ArcView window.
- Close -- associates as script that executes whenever the dialog box is closed
- Closeable -- indicates whether or not a dialog can be closed using the window frame controls. The controls are disabled when the Closeable property is FALSE.

- EscapeEnabled -- When TRUE, allows the user to dismiss the dialog box by pressing the ESCAPE key.
- HasTitleBar -- determines whether or not the dialog window will have a title bar. The frame will be titled with the text set in the Title property if HasTitleBar is TRUE.
- Modal -- prevents the user from working with any function in ArcView other than the dialog if TRUE.
- Open -- the script that runs whenever the dialog is opened.
- Resizable -- allows the user to resize the dialog box if set to TRUE.

List box property settings for the PGA Update dialog box:

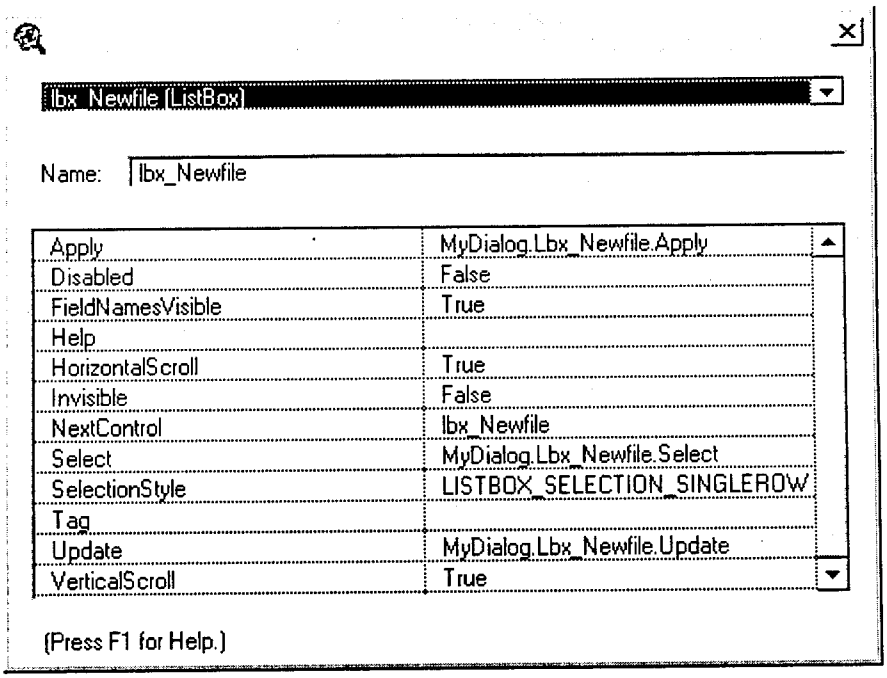


Figure 3.5: PGA Update dialog box list box properties

- Apply -- the script that executes when an *apply* event occurs. In the case of lbx_Newfile, the apply event occurs whenever a record inside the listbox is double-clicked.
- Disabled -- determines whether or not the listbox will be disabled. The listbox is disabled if this property is set to FALSE. Controls that are disabled are typically enabled when their Update script is run.

- FieldNamesVisible -- when the list box control is derived from a VTab, the field names will appear above the listbox if the FieldNamesVisible is set to TRUE.
- HorizontalScroll -- determines whether or not there will be a horizontal scroll bar in the list box. If FALSE, no horizontal scroll bar will appear and the elements inside the list box are sized to fit.
- Invisible -- sets the control invisible if TRUE. Controls that are invisible are typically set visible by the SetVisible request inside their Update script.
- NextControl -- specifies the control that will receive keyboard focus when the TAB key is pressed.
- Select -- name of the script that executes when a record is selected in the list box.
- SelectionStyle -- an enumeration that defines what type of selection is made from the list box. Lbx_Newfile is set to select a single row.
- Vertical Scroll -- determines whether or not there will be a vertical scroll bar in the list box. If FALSE, no vertical scroll bar will appear and the elements inside the list box are sized to fit.

Label button properties for the PGA Update dialog box:

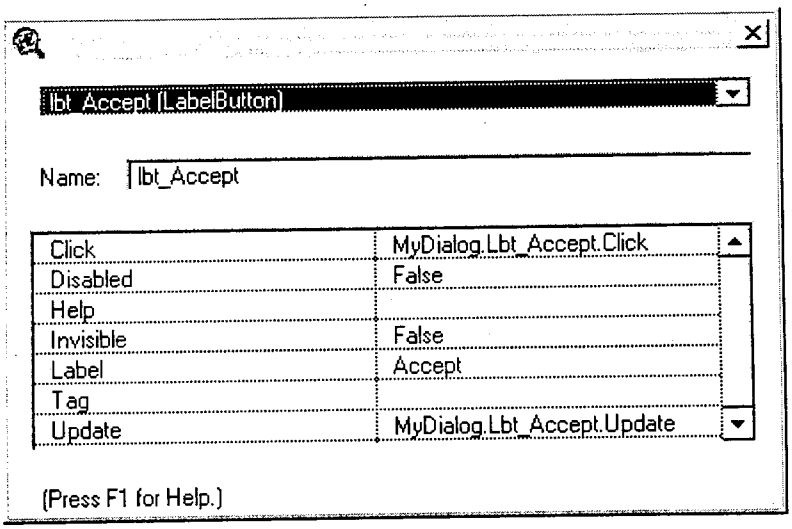


Figure 3.6: PGA Update dialog box Accept label button properties

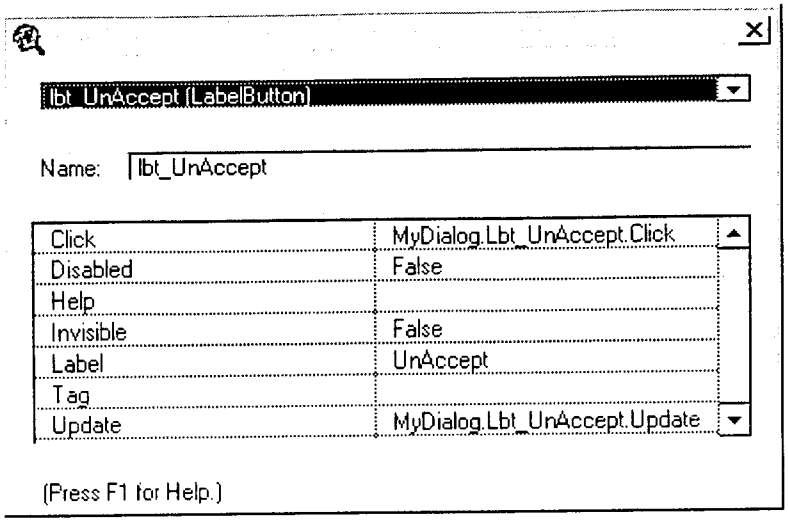


Figure 3.7: PGA Update dialog box Unaccept label button properties

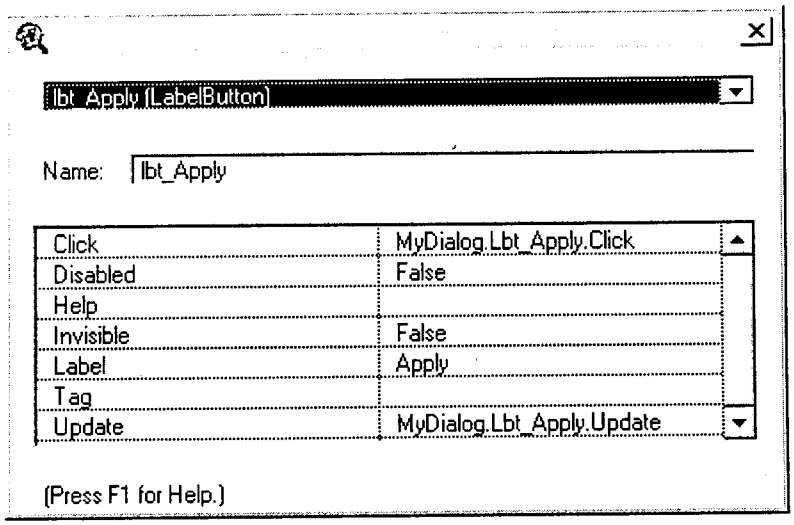


Figure 3.8: PGA Update dialog box Apply label button properties

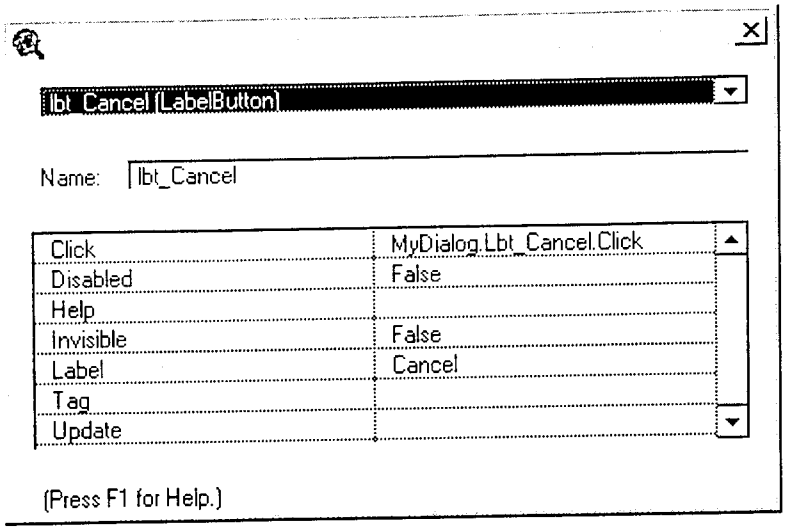


Figure 3.9: PGA Update dialog box Cancel label button properties

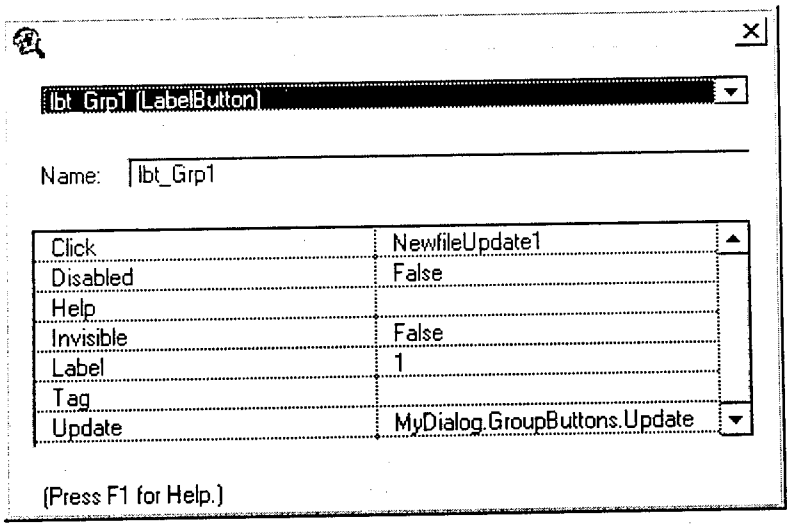


Figure 3.10: PGA Update dialog box Group 1 update label button properties*

* The *Update Group Assignment* label button properties are similar for all seven buttons. The Label field values for the seven buttons are 1, 2, 3, 4, 5, 6, and 7. These labels reflect the values that are calculated into the selected record's PGA when the button is clicked. The Click properties are associated with the NewfileUpdate'x' scripts, where 'x' is equal to the label text (i.e. the Click script for label 2 is NewfileUpdate2).

- Click -- name of the script that executes whenever the label button is selected or "clicked".
- Disabled -- indicates whether or not the contol is disabled. By default, Disabled is FALSE. When set to TRUE, the button is "greyed" out.

- Invisible -- sets the control invisible if TRUE. Controls that are invisible are typically set visible by the SetVisible request inside their Update script.

- Label -- the text that appears on the label button

- Update -- the script that runs whenever an Update event occurs. Update events occur whenever a broadcast is sent to the listener component. For this program, the listener and broadcaster controls are set in the script MyDialog.Open.

Once a site or group of sites is selected, either by spatial or logical means, the dialog box can be opened from either the View or Table GUI by clicking the Open PGA Update Dialog Box button located on the GUI button bar.

 Open PGA Update Dialog Box button

This executes a script associated with the dialog's open property, which, among other things, creates a temporary table called *newfile* whose VTab is imported from the selected records of the PTC attribute table. These records are used to populate the list box in the PGA Update dialog. *Newfile* only exists as long as the dialog box is open; when the dialog is closed the values from *newfile* are used to edit the PTC inventory table and *qtable*.

While open, the dialog box gives the user access to various controls that will aid in editing the PTC inventory table. The following paragraphs describe each control and its function in detail.

List box (Lbx_Newfile). The list box values (fields and records) are loaded from *newfile*. The records can be selected one-at-a-time, which sends a broadcast to the Accept and Unaccept label button controls. This runs their update scripts that dictate which button will appear on the dialog box. If the AcceptFlag field for the record is zero (0), then the Accept button will appear. Otherwise, the Unaccept button is activated.

While open, the list box displays the updated records of *newfile* as they are edited. No edits are actually made to the PTC inventory table or *qtable* until the Apply button is pressed. When double-clicked, a record's PGA value is automatically calculated to be equal to its Best_Fit1 value, and the AcceptFlag is calculated to reflect the current date.

Accept and UnAccept label buttons (Lbt_Accept, Lbt_UnAccept). Only one of these buttons are available at a time. Initially both are inactive, but when a record inside the list box is selected, either the Accept or UnAccept buttons becomes available. The two buttons are overlaid on top of one another and act as a toggle. If the selected record's AcceptFlag value is zero (0), then the Accept button is activated. Else, the UnAccept button is promoted to the top, hiding the Accept button, and it becomes active.

When clicked, the Accept label button executes the script MyDialog.Lbt_Accept.Click which calculates the current date as an integer for the selected record's AcceptFlag field. At this point, the Accept button becomes inactive, and the UnAccept button promotes to the top of the dialog and is activated.

If the UnAccept button is clicked, the selected record's AcceptFlag value is recalculated to zero (0) and the Accept button promotes and becomes active.

Apply label button (Lbt_Apply). No edits are made to the PTC inventory or *qtable* until the Apply button is clicked. When executed, the associated script writes the edited records from *newfile* to the PTC inventory table in Access via a series of SQL commands. The edits are also made to *qtable* so that the changes are visible in the PTC theme (recall that *qtable*'s Classify field dictates how the PTC sites are symbolized in the SGAA view).

Once the tables are edited, the temporary table *newfile* is deleted and the dialog box closes.

Cancel label button (Lbt_Cancel). The cancel button dismisses the dialog box without making any edit changes to the PTC inventory table or *qtable*.

Update PGA Group buttons (labeled 1 through 7). These buttons calculate the list box's selected record's PGA field. Numbered '1' through '7', the labels indicate what value is calculated into the field. These calculations are made to the table *newfile*, which in turn is used to update the PTC inventory Access table when the Apply button is clicked. In addition to calculating the PGA, whenever an update group button is clicked the AcceptFlag is also calculated reflecting the current date.

3.1.4 ACCEPTING SITE PGA VALUES FROM THE VIEW GUI

The program provides the user a means of calculating the AcceptFlag for a selected site or sites without opening the PGA Update dialog box. The analyst is able to determine what the PGA value is for a site by noting its symbology in the SGAA View. If the value is acceptable, the user can select the site (or sites) and calculate the AcceptFlag for both the PTC inventory table and *qtable* by clicking the Accept PGA button located on the View GUI button bar.

Accept PGA button

3.2 HOT LINKS FOR SITE SAS JMP OUTPUT IMAGES

Another function of the program enables the user to view the seasonal profile graphs and each station's SAS JMP output. The former was done by associating the images to a drop down menu item in the View GUI (see section 3.3). An established hot link between the PTC sites and their individual images provided the latter. An *on-the-fly* method is used to establish the hot link between the features in the view and their associated image. This

system creates the link whenever a site is selected with the appropriate hot link button located on the View GUI tool bar.

 Site SAS Analysis Image Hot link button

The images are stored in a subdirectory named 'htimages'. These files are named with the following convention:

Station(CovSysID).gif

Where '(CovSysID)' is the site's unique identifier. When a site is clicked, the tool's associated script creates a relative path and filename, locates the GIF, and displays the image in an image window. The filename is created by concatenating the CovSysID of the chosen site with the prefix 'station' and suffix '.gif'. The following is an excerpt of the script MyView.Hot link (refer to *Suppliment A: Script Documentation*):

Where theField is CovSysID

```
...
      theVal = t.ReturnValueString(theField.GetName, rec)
      if (not (theVal.IsNull)) then
         theVal = "c:\seasonal\htimages\station"+theVal+".gif"
          if (File.Exists(theVal.AsFileName)) then
           i = ImageWin.Make(theVal.AsFileName, theVal)
           i.SetScaled(false)
           i.Open
           i.Resize(650,700)
           i.MoveTo(375,50)
         else
           MsgBox.Warning("File "+theVal+" not found.","Hot Link")
         end
      end
...
```

This procedure eliminated the need to store an image path and filename inside the PTC attribute table. The advantage is a smaller table and an easier approach to moving the images. Also, no editing of records in the table is needed when images are created for the PTC sites.

3.3 SEASONAL PROFILE IMAGES

The ATR Graphs button was added to the View GUI button bar after a suggestion was made by a member of the program staff. The button opens a single image that contains all seven seasonal profiles.

 ATR Graphs button

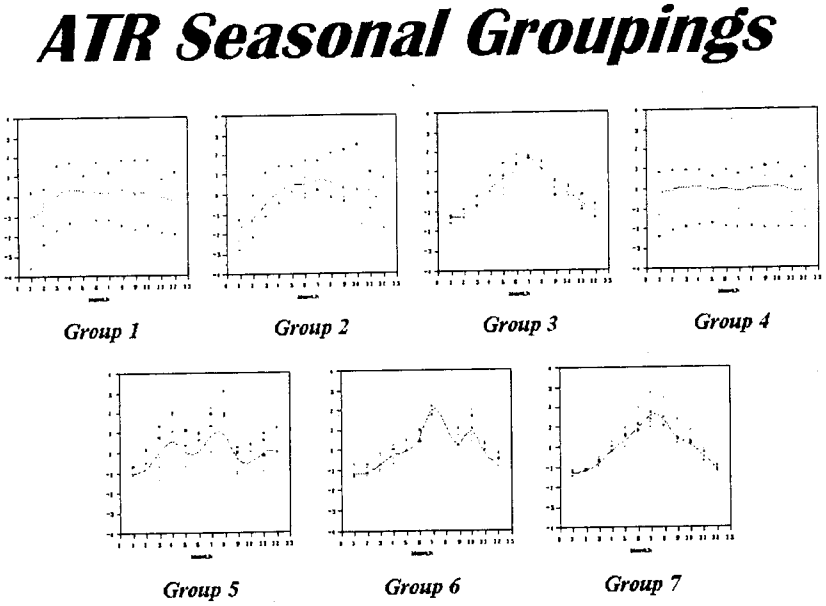


Figure 3.11: Seasonal Profiles for Groups 1 through 7

Each profile can be viewed independently as well. A menu button was added to the View GUI and named "ATR Groups." This menu contains seven menu choices, each named after the profile image they launch (e.g. Group 1, Group 2, Group 3, etc.).

3.4 VISUAL GUIDANCE: INTERACTIVE LOCATOR MAP AND ANNOTATION

The ability for the analyst to keep track of where he or she is while working in the ArcView interface is essential to optimizing efficiency in editing the PTC inventory table. Annotation, or labeling, is necessary to providing the ability to identify a feature or features on a map or view. Also, a locator map provides an effective way in which a user can quickly find the area he or she is working. Both of these concepts were implemented in this customized ArcView program.

Annotation for the ATR sites was originally produced manually and stored as attached graphics in the ATR theme. However, the creation of these graphics proved to be a long process. It was decided by the project staff to use the auto-labeling mechanism supplied by ArcView. Although the aesthetics are crude, the flexibility and speed of auto-labeling features is a more effective method. Also, the program's auto-labeler can be used to label the features of a theme with any of its values in the theme's attribute table.

Another feature added to the program labels the PTC stations in the SGAA View when the PGA Update dialog box is open. The function was requested by the program staff after the initial trial of the beta program. While working inside the PGA dialog box, difficulty arose when the analyst tried to visually link the features with their records in

the list box. By annotating the PTC sites with values from an attribute table's field (set to CovSysID by default), the analyst can easily recognize a feature in the SGAA view with its associated record inside the dialog box. When the dialog is dismissed, the labels are selected and deleted. This is a quick process, since only the PTC features within the view's extent are labeled.

By clicking the Locator Map button (located in the View GUI button bar), the user can open an *interactive* locator map view.

 Locator Map button

This view is composed of one theme depicting North Carolina's county boundaries, several zoom buttons and tools, and two custom buttons. When open, the locator view displays the county boundaries of North Carolina and a rectangle graphic showing the extent of the SGAA View. The map is interactive in that the user can resize or move the graphic to change the extent in the SGAA View by clicking and dragging the box by one of its handles and clicking the Change SGAA View Extent button.

 Change SGAA View Extent button

Another option to the user is to return to the SGAA View without changing the extent. This can be accomplished by clicking the customized Back to SGAA View button.

 Back to SGAA View button

Attachment A - Recommendations from the NCSU Department of Statistics

Dr. Johnson's Recommendations for Assigning Groups

Assigning a Station to a Seasonal Group

It will usually be best to assign a Station to the Seasonal Group with which its counts are most highly correlated over days and months. To help with the visual comparisons, compare the traffic counts at a Station with the inverse of the Index for each Seasonal Group. This makes both counts and Indexes begin with relatively low values in January and increase to relatively high values during the summer. Standardize the values of both the Indexes and the Counts and put the results on a common scale in plots of the values versus Month. Standardize the values by subtracting the mean and dividing by the standard deviation. This transformation is often called the Z transformation.

Although it is usually best to assign a Station to the Seasonal Group with which the readings are most highly correlated, several groups have similar profiles and there may be other reasonable choices. Practice with a few stations has shown that two or three correlations may be very close. This means that with good judgement, an engineer may use the assignments of neighboring stations to choose among the reasonable choices for Seasonal Group to which to assign a given station. We do not have a formal automated procedure to make such comparisons. However, the information contained in the recommended reports for each station will support the informed engineering judgement for such assignments.

Sampling Traffic Counts at Stations as the Basis for Assigning a Group

When the Group for assignment is clear from past history or the surrounding stations, data from three or four 48-hour sampling periods might be sufficient to estimate the scaling multiplier. The scaling multiplier is the number by which the Seasonal Group Index is multiplied to get the estimated profile for the station. However, when the Seasonal Group assignment is in doubt, additional 48 hour sampling periods may be required to have reasonable confidence in the assignment.

Refer to the figure Inverse Seasonal Index Patterns by Month to see differences in patterns that you are trying to find with the sampling procedure. See the table Inverse Seasonal Index Pattern by Month for the Pairwise Correlation among the seven Seasonal Groups. Profiles of Groups 6 and 7 are highly correlated (0.9635), which means it would take a lot of observations to determine whether a Station should be assigned to Group 6 or to Group 7. However, because Groups 6and 7 are so highly correlated, this distinction should not be critical. Groups 3 and 7 are also highly correlated (0.9342). The main difference between Group 3 and Group 7 seems to be that there is more variation between days in Group 3. Therefore, readings for entire weeks would be required to distinguish between Groups 3 and 7.

Groups 1 and 2 are highly correlated (0.9387) with the main distinction being more of a drop in September for Group 2 than for Group 1.

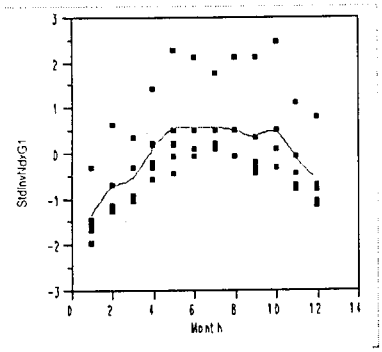
Groups 4 and 5 have visually unique profiles and are not highly correlated with any other Group. The distinctions for Group 4 are the drop between April and May, the bimodal pattern (with peaks in April and September) and the great variation between days of the week in the last half of the year. The distinctions for Group 5 are the drop from August to September, the increase from September to December and the drop from December to January and February.

To find these distinctive differences, it would be well to begin with a 48-hour sampling period in each of the even numbered months 2, 4, 6, 8, 10 and 12. To measure the variation between days of the week, a full week of readings should be included in month 6 or month 8. To identify groups 3,4 and 5, additional sampling periods should be added in months 5, 7 and 9. If these results do not establish an assignment with sufficient confidence, additional sampling periods should be assigned to the months that would most reduce the uncertainty in assignment.

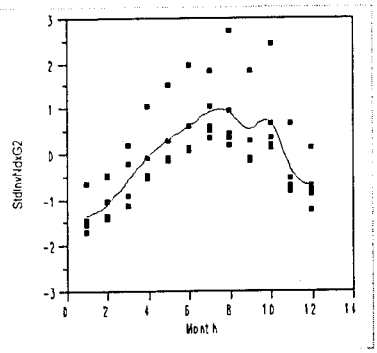
Inverse Seasonal Index Pattern by month

(fit with spline: lambda = 0.1)

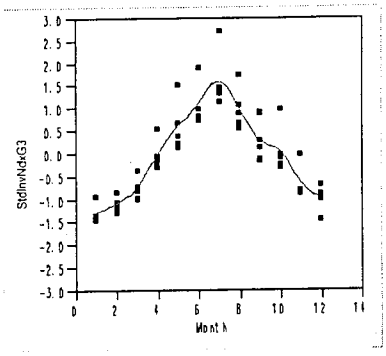
Group 1



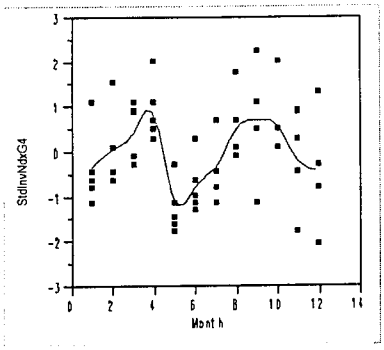
Group 2



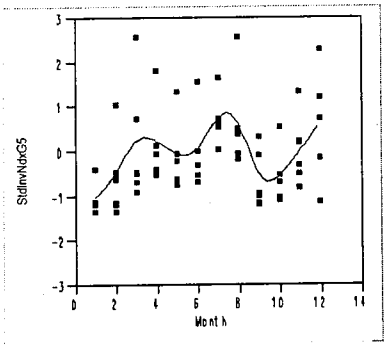
Group 3



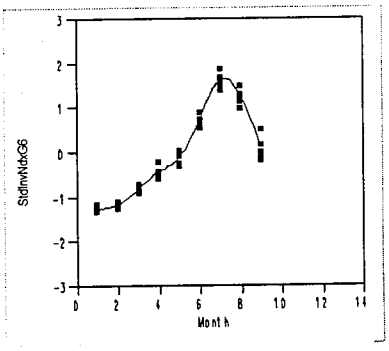
Group 4



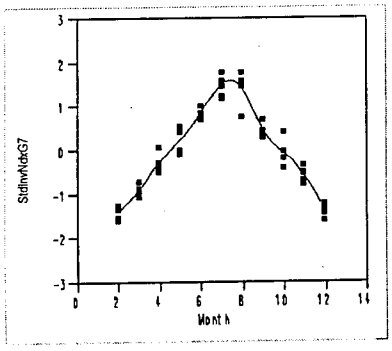
Group 5



Group 6



Group 7



Inverse Seasonal Index Pattern by month

(fit with spline: lambda = 0.1)

Pairwise Correlations

Variable	by Variable	Correlation	Count	Signif Prob
InvNdxG2	InvNdxG1	0.9387	60	0.0000
InvNdxG3	InvNdxG1	0.7764	60	0.0000
InvNdxG3	InvNdxG2	0.8834	60	0.0000
InvNdxG4	InvNdxG1	0.5602	60	0.0000
InvNdxG4	InvNdxG2	0.4595	60	0.0002
InvNdxG4	InvNdxG3	0.1544	60	0.2389
InvNdxG5	InvNdxG1	0.6715	60	0.0000
InvNdxG5	InvNdxG2	0.6152	60	0.0000
InvNdxG5	InvNdxG3	0.4832	60	0.0001
InvNdxG5	InvNdxG4	0.4862	60	0.0001
InvNdxG6	InvNdxG1	0.6058	45	0.0000
InvNdxG6	InvNdxG2	0.7826	45	0.0000
InvNdxG6	InvNdxG3	0.9180	45	0.0000
InvNdxG6	InvNdxG4	-0.0179	45	0.9073
InvNdxG6	InvNdxG5	0.4659	45	0.0013
InvNdxG7	InvNdxG1	0.5781	55	0.0000
InvNdxG7	InvNdxG2	0.7648	55	0.0000
InvNdxG7	InvNdxG3	0.9342	55	0.0000
InvNdxG7	InvNdxG4	0.0573	55	0.6778
InvNdxG7	InvNdxG5	0.2533	55	0.0621
InvNdxG7	InvNdxG6	0.9635	40	0.0000

Attachment B - Justification request for rewording of Project Task 2

AS REPORTED FOR THE: QUARTERLY REPORT, SEPTEMBER 30, 1998

July 1, 1998 – September 30, 1998

Prototype Demonstration of a Geographic Information System Application For the Seasonal Analysis of Traffic Data, Development of Seasonal Factors And Seasonal Adjustment of Roadways

Justification request for rewording of Project Task 2 (approved)

Subject: Highway Research Project 99-5

"Prototype Demonstration of a Geographic Information System Application for the Seasonal Analysis of Traffic Data, Development of Seasonal Factors and Seasonal Adjustment of Roadways"

Project Authorization To: The Institute for Transportation Research and Education (ITRE)

Proposal for modification of task methodology

NCDOT Project Monitor David White, P. E., ITRE Principal Investigator Jay Novello, and NCSU Project Advisor Dr. Tom Johnson (herein referred to as "the core project group"), recommend that one item in the Project Proposal undergo a wording change, based on research conducted under the terms of the project to date. We are presenting our findings and requesting comments from the Project Technical Committee prior to proceeding in the manner outlined in this message.

In the outline of the Research Methodology and Tasks in the Project Proposal, Task 2 specifies the development of an application to assign a "best guess" seasonal group to each short-term traffic count site in the pilot area, based on the most recent readings at that site. In the detail item "c" for this particular task, it is stated that "the application will be run as an unattended process, and no user input will be required". The core project group recommends deleting this sentence from the project methodology.

The statistical analysis software considered for project task 2 included Microsoft Excel, Systat, SPSS, ARC/INFO, SAS JMP and "Big" SAS. Excel and ARC/INFO were eliminated from consideration early in the project research due to their lack of core statistical analysis and visualization features. Excel will do basic statistics, but it is not

suitable for exploratory comparisons that seem to be important to assign a station to a Seasonal Group. Systat and SPSS are expensive packages and present a steep learning curve. Consideration then focused on "Big" SAS and SAS JMP. The core project group has identified the following key points about each of these packages:

"Big" SAS:

- Can be programmed to run in unattended "batch" mode
- NCSU Statistics Consulting Group can be contracted to write program
- Output primarily in tabular form
- More difficult to apply informed engineering judgment for group assignments (almost a "Black Box" approach)
- Licensing cost extremely high
- Large program such as "Big" SAS will be required if all 50,000+ stations are analyzed at once

SAS JMP:

- Interactive data-exploring interface
- Easier to investigate the integrity of the data
- Methodology for determining "best guess" for each count station already researched and demonstrated by Dr. Johnson, who will document this approach in "cookbook" format under terms of the current project
- Feasible for projects assigning a few hundred stations at a time
- Output in graphical or tabular format
- Supports informed engineering judgment in group assignment
- Current version cannot be programmed to run in "batch" mode, but upcoming beta version (3Q 1998) promises this functionality
- Purchase price affordable (approximately $800)

Based on this information, the core project group recommends using an existing user driven, interactive software package versus developing and programming a fixed, inflexible statistical batch program.

Task 2 requires the development of a statistical application
that assigns a "best guess" seasonal group to a short term count site. This requirement will be met using the user-interactive SAS JMP Statistical Discovery software. This approach is considered to be superior to the non-interactive unattended batch mode.

Dr. Johnson will develop and document a step by step statistical methodology and procedure using the SAS JMP Statistical Discovery software.

The results from SAS JMP will be data input to the GIS application as outlined by Task 3. Task 3 and the functionality of the GIS application has not been changed nor modified by this new statistical analysis approach.

Before we proceed and continue to develop this proposed methodology, the members of the Technical Advisory Committee are solicited to provide any comments, suggestions, or alternative proposals to Jay Novello and David White by September 8th, 1998. Members of the Technical Advisory Committee who do not respond to this solicitation will be considered as concurring with the new statistical "best guess" approach. If additional information is required, please contact David White. After reviewing the comments from the Technical Advisory Committee, the core project group will address the T. A. Committee comments and attempt to finalize a scope for Task 2.

Thank you for your attention to this matter.

David White, P. E. (dwhite@tpswp01.dot.state.nc.us)
Dr. Tom Johnson (tom_johnson@ncsu.edu)
Jay Novello (jay@itre.ncsu.edu)

Attachment C

Steps to get a Station Data Table and Report for a New Station Called Station # in the instructions

Open the data tables that you will use.
File → Open → Station009.jmp →File → Open → SeasoNdx.jmp →
File → Open → ContSite.jmp →

In the data table **ContSite.jmp** select the rows for Station #.
Rows → Select → Where... → Station **→ equals [▾] →** # **→ [OK] →**

Using Ctrl-Click, highlight columns **Station** and **Avg. Day→**
Tables → Subset You will get an Untitled table with 60 Rows and 2 Columns
Window → SeasoNdx → Tables → Join → Untitled_ → By Row Number →
Output Table: Station# **→ [Join]**
Data Table Station# will have 60 Rows and 11 Columns. →

File → Save → Select folder in which to Save **→ Save →**

Rename column **Avg. Day** to **Avg.dayS# →**

Get the formula to standardize **Avg. DayS#.**
→ Window → Station009 → Double-click in the header box of **StdSta009.**
→ Click on the Formula in the lower left-hand corner of the Column Information window. → Click on the fraction bar in the formula to highlight all of the formula.
→ Edit → Copy → Close the Formula window **[X]. → Window → Station# →**

Create a new column standardizing **Avg. DayS# → Cols → New Column... →**
Col Name: StdSta# **→ Data Source: Formula [▾] → [OK] → Edit → Paste →**
choose **Avg. DayS# →** Close Formula window **[X]. → File → Save →**

Plot **StdSta#** against **Month.**
Window → Station# → Analyze → Fit Y by X → Month → >>X>> → StdSta# →
>>Y>> → [OK] → Fitting [▸] → Fit Spline → .1 →

Change the range of the Y axis to be –3 to 3 with steps of 1 and no decimals. →
Double-click on a number on the **StdSta#** axis.**→ Minimum: -3 →**
Maximum: 3 →Increment: 1 → Minor Ticks: 0 →
Format: Fixed Decimal [▾] 0 → [OK] →

Copy the graph to the Windows clipboard.
→ Choose the Scissors tool. →
Alt-Click on (StdSta# by Month) **→ Edit → Copy →**

Paste the graph into the top part of the report on Station #.

Create the Pairwise Correlation matrix of Season Group Indexes with Avg. Day#.

Window → Station# → Analyze → Correlation of Y's → InvNdxG1 →
>>Add>> → ... → InvNdxG7 → >>Add>> →
OR, click-and-drag to highlight InvNdxG1 through InvNdxG7, the **>>Add>>**.
Avg. DayS# →>>Add>> →[OK] →
Border Check-Mark [✓] → Correlation-Pairwise →

Copy the Pariwise Correlation table to the Windows clipboard.
Scissors Tool →
Alt-Click (Pairwise Correlation) **→ Edit → Copy →**

Paste the table into the bottom part of the report for Station #. To make the table fit, resize the table to 50%, crop the picture to cut off the histogram and top part of the table, then resize back to 100%. Note that the column headings are included in the template.

If you want to add the bottom row of the Scatterplot matrix to your report, take the following steps:

[✓] → Scatterplot Matrix → Scissors Tool →
Alt-Click (Scatterplot Matrix) **→ Edit → Copy →**
Paste the Scatterplot matrix into the bottom part of the report for Station #. Crop the picture to cut off the top seven rows of scatterplots. Note that the labels for the horizontal axes are included in the template.

Close all of the Windows in JMP that you will not use with the next station. Wait to close the JMP windows until you are sure that you will not need to re-do the copy-and-paste into the report.

Note: You can recreate the formula to standardize **Avg. DayS#** with the following steps:

Window → Station# → Cols → New Column... → Col Name: StdSta# **→**
Data Source: Formula [▾] → [OK] →
Key Pad: (→ Column List: Avg. DayS# → Key Pad: - →
Function Type: Statistical → Function: Mean →
Column List: Avg. DayS# →Click on **) → Key Pad: ÷ →**
Function Type: Statistical → Function: Std Deviation →
Column List: Avg. DayS# → Close Function window **[X]**

Format for Season Inverse-Index Data Table

Day Code	Month	InvNdxG1	InvNdxG2	InvNdxG3	InvNdxG4	InvNdxG5	InvNdxG6	InvNdxG7
2	1	0.900901	0.78125	0.581395	1.041667	0.704225	0.164745	.
2	2	0.934579	0.8	0.625	1.06383	0.806452	0.186916	0.301205
2	3	0.961538	0.833333	0.680272	1.086957	0.793651	0.384615	0.52356
2	4	0.990099	0.925926	0.884956	1.111111	0.909091	0.571429	0.813008
2	5	1	0.980392	1.123596	0.990099	0.877193	0.869565	1.162791
2	6	1.041667	1.010101	1.162791	1.020408	0.862069	1.25	1.282051
2	7	1.052632	1.06383	1.298701	1.06383	1.020408	1.754386	1.492537
2	8	1.030928	1.052632	1.086957	1.098901	0.990099	1.612903	1.315789
2	9	1	1.030928	1.020408	1.020408	0.900901	1.020408	1.111111
2	10	1.010101	1.041667	0.917431	1.098901	0.819672	.	0.970874
2	11	0.980392	0.877193	0.719424	1.06383	0.769231	.	0.649351
2	12	0.970874	0.877193	0.70922	1.075269	1.136364	.	0.458716
3	1	0.877193	0.75188	0.555556	1.020408	0.671141	0.141643	.
3	2	0.934579	0.793651	0.598802	1.06383	0.671141	0.188324	0.325733
3	3	0.952381	0.862069	0.675676	1.075269	0.75188	0.411523	0.552486
3	4	1.010101	0.917431	0.862069	1.123596	0.819672	0.588235	0.78125
3	5	1.030928	0.980392	1	0.980392	0.78125	0.775194	0.990099
3	6	1.041667	1	1.136364	1.010101	0.793651	1.351351	1.282051
3	7	1.041667	1.041667	1.25	1.06383	0.925926	1.923077	1.515152
3	8	1.030928	1.020408	1.123596	1.086957	0.884956	1.818182	1.612903
3	9	1.020408	0.980392	0.909091	1.123596	0.735294	0.813008	1.123596
3	10	1.010101	1.010101	0.877193	1.098901	0.719424	.	0.900901
3	11	1	0.884956	0.70922	1.06383	0.826446	.	0.684932
3	12	0.952381	0.869565	0.694444	1.041667	0.892857	.	0.362319

Format for ContSite Data File

Group	Station	Day Code	Month	Avg. Day
1	1	2	1	.
1	1	2	2	.
1	1	2	3	4930
1	1	2	4	4732
1	1	2	5	4543
1	1	2	6	4487
1	1	2	7	4589
1	1	2	8	4752
1	1	2	9	4418
1	1	2	10	.
1	1	2	11	4638
1	1	2	12	4445
1	1	3	1	.
1	1	3	2	.
1	1	3	3	4819
1	1	3	4	4763
1	1	3	5	4855
1	1	3	6	4311
1	1	3	7	4370
1	1	3	8	4574
1	1	3	9	9646
1	1	3	10	.
1	1	3	11	4481
1	1	3	12	4688

Attachment D - Alpha Version Application Creation Procedures

1. Incorporating data into the ATR theme attribute table from SAS statistical software reports

   The shapefile atr.shp is a point data source depicting each ATR station's location by an x,y -coordinate. The accompanying database, atr.dbf, contains attribute information about each site, including, among other things, the ATR unique ID, station number, route, and location. New information obtained through the SAS software needed to be incorporated into the database as well. This was accomplished by creating a new Dbase table, hot linktable.dbf, with three attributes named *best_fit1, best_fit2,* and *best_fit3* depicting the three best-fit groups for each ATR station. A fifth attribute, *current_assignment*, is a copy of the values in *best_fit1*. This field is used to symbolize the ATR theme, and will be updated by the user if necessary with values from *best_fit2* and *best_fit3*. The ATR unique ID was also added to the new table to act as a link between it and atr.dbf.

2. Creating the project and organizing data

   The project, seasonal.apr, was first customized to contain two View components. The first acts as an interactive map display of the data, displayed in a View as themes, and the other a locator map.

   The first View component contains one view window, entitled NCSU-ITRE Seasonal Group Assignment Application, or SGAA. The SGAA view displays several statewide themes, including county and municipal political boundaries, ATR stations, and various surface features.

   Once the atr.shp shapefile was added to the SGAA view it was renamed Continuous Traffic Count Station and its attribute table was opened. The table was joined to hot linktable.dbf. With the information from hot linktable.dbf accessible through the attribute table, the theme was categorized into eight classifications based on each station's (i.e. *current_assignment* = 0), it was keyed as an eighth classification, *Not Enough Data*.

   Other themes were added to the SGAA view. If appropriate, the theme's name and symbol were changed. The following list indicates the themes added by their name and the source:

NAME	SOURCE	SOURCE TYPE
Continuous Traffic Count Station	atr.shp	shapefile
Primary Road	lrs96 (route.hwy)	cover
Secondary Roads	nonlrs (arc)	cover
Railroad	rail (arc)	cover

Surface Water	hydro.shp	shapefile
Municipality	mb1997 (region.towns)	cover
North Carolina County	cbsl (region.co)	cover

Some of the coverages have a minimum and maximum scale in which they will display. This prevents too much clutter when viewing the display at a small scale. The Secondary Road theme, for example, has a maximum display scale of 1:250001. Thus, its features do not display when the view scale is smaller than 1:250000. The minimum and maximum scales at which a theme displays is manipulated by the theme property's *Display* component.

1. Creating Annotation

Four annotation layers were created and added to the SGAA view by converting a few features from the cbsl shapefile into a new shapefile, label_extent.shp. This shapefile was in turn copied three more times and renamed ATR Annotation Layer 1, 2, 3, and 4, respectively. Each ATR station was then labeled at a small scale. These labels were attached to ATR Annotation Layer 4. The mapextent was then scaled up and each ATR station was labeled again. Those labels were then attached to ATR Annotation Layer 3. This process continued twice more until there were four complete annotation layers, all depicting the ATR unique ID, but at different font sizes. Each layer was scaled so when a user zooms into or out from the display, the size of the annotation will grow or shrink as well. To omit overlap, each layer has a specific minimum and maximum scale in which it will display. For example, the layer ATR Annotation Layer 3 displays when the display is scaled between 1:550000 and 1:850000. Once the display is zoomed in past 1:550000, Layer 3 no longer displays. Layer 2 has a minimum / maximum scale between 1:300000 and 1:550000, so when Layer 3 turns off, Layer 2 turns on. This prevents labels from becoming too large when a display is shown at a large scale and too small at a small scale.

A fifth annotation layer was created for labeling some of the primary highways. This was done at a small scale extent, and only for Interstate and US highways. At a scale larger than 1:550000 or smaller than 1:850000, the labels do not appear.

2. Creating a locator map and associated scripts

The second of the two view components was added to incorporate a special view, named Locator Map, with a customized interface. The view component was named Locator and given a unique icon. An Avenue script called View.LocatorMap associated with a button on the SGAA view opens, activates, and promotes the Locator Map and draws a box mocking the extreme coordinates of the SGAA view extent. The size and position of the box can be manipulated by the user by clicking and dragging the edges. Another script, named View.ExtentRedraw and associated with a button labeled "E" in the Locator Map view interface, allows the user to return to the SGAA view redisplayed to the extent of the resized / repositioned locator box. A third button, labeled with a "B", is associated to a simple script,

View.BackToView1, that returns the user to the SGAA view at its extent previous to activating the Locator Map View window.

Each of the three scripts were written in ArcView's script editor and are accessible through the project's Scripts component.

3. Creating a hot link

Each ATR feature has an accompanying document that contains statistical information in graphs by group for that station. The documents were created using Microsoft Word. An image of the .doc files were captured and stored as .gif files, each named by according to their unique ID. For example, the document station0001.doc was captured as station0001.gif. These .gif images were stored in their own directory. Then a new character field was created in hot linktable.dbf called *images*. Values indicating the location of each station's associated .gif image were calculated into the field. A hot link was then activated for the Continuous Traffic Count Station theme. The hot link is activated through the theme property's *Hot link* component.

4. Creating the splash-screen banner

The initial startup of the project from ArcView takes a moment to load. During that time interval, a banner introducing the project and another that reads *Project Uploading, Please Wait...* displays. These banners, or message boxes, are activated using a startup script that remains active for twenty seconds. This period is based on the time the project takes to start on a local PC with a 200 Mhz processor and 32Mb of RAM.

Attachment E - Beta Version Design Notes

These subjects were discussed by the core project group at its November meeting.

ATR site annotation – can its creation be automated?

The creation of ATR site annotation done by hand using a new Arcview 3.1technique. We can either continue to use this technique or create it automatically; auto-created annotation will scale properly, but as view is zoomed in the annotation moves farther and farther away from the location of the point it annotates. There is no perfect solution to this well-known Arcview problem.

Hotlinking of ATR sites to JMP analysis document

We can't figure out a way to un-check the "scale image" box in the popup window; as of now, this has to be done by hand. However, the size of the window can be specified in Avenue. However, is a .gif screen capture the appropriate format to use? Could a DDE call from Arcview be used to link to a "live" MS-Word document? Would this allow the hotlink to be used for something else, like an image? What if we converted the MS-Word document to PDF, and use the hotlink to summon that document in a PDF viewer?

New button requested on interface

We need a button on the interface somewhere that brings up a document with the seven seasonal "average" profiles, and maybe some other information; DOT has taken the responsibility to create this document.

Terminology change

In the existing application, change references to "not enough data" to "not assigned".

Development of project documentation

The documentation should evolve in two parts -- the "user-level" cookbook for the technician in charge of examining the PTC site assignments, and the "administrator-level" cookbook for project
setup and shutdown functions. (Most of the latter will be Access functions and as such will be developed and documented by NCDOT.)

The documentation should include a Data Flow Diagram listing software modules, direction and content of data flow, and any custom scripts, documentation, and table joins used. The supporting documentation should also include a listing of scripts (well-commented for the benefit of future DOT code maintainers) and their associated GUI elements, and an explicit listing of any fields/themes/data
sources they refer to directly.

Application training

NCDOT Arcview/application training is deferred until the US 70 pilot corridor data capture is completed, analyzed, and incorporated into the application and documentation. This will result in a substantial change to both, so it was agreed that is was best not to go through this exercise twice. When the pilot corridor data is delivered, ITRE will do the SAS analysis, rebuild the training documentation, and return the data back to DOT to process on their own.

Implementation of the ATR/PTC data within MS-Access

Project Technical Committee member Julia Harrell is working on the ODBC link between Arcview and MS-Access. Others have been successful in setting up this link. DOT wants Access to be at the center of this application (not a .dbf table) since the ATR and PTC data is already stored that way.

The Access table of all 65,000 PTCs will be the basis for the final version of the application. The matching shape file of PTC site locations will be created gradually, and newer and more complete versions of this file will be substituted in the applications as they arrive from the digitizing shop.

The Access table will include these fields for each PTC station:

its current group assignment

its date stamp, indicating the last time the current group assignment was made official

an "Access Flag", which will be raised when a SAS JMP analysis has been performed on the station.

fields for "Best fits" 1 through 3, which will be populated with seasonal group candidates from SAS JMP for those stations whose "Access Flag" is raised.

the "pending" assignment for that station. For stations whose "Access Flag" is not raised, this will be calculated to be the same as its current group assignment. For stations whose "Access Flag" is raised, it will be calculated to be the same as "Best Fit 1".

an "accept" flag. When the table is set up by the administrator, none of the "accept" flags will be raised.

The procedure of building this table will be known as a "reset", and will be performed once a year by the application administrator (see below).

Routine data processing procedure

The analyst in charge of actually operating the application (heretofore referred to as the "user") will make two passes at the data. The first pass will be to approve or change the "pending" assignment for stations for which SAS JMP analysis has been completed (that is, the Access Flag is raised and the "pending" assignment has been calculated from "Best Fit 1"), based on the proximity of the seasonal classifications of neighboring stations and ATRs.

The second pass will be to approve or change the "pending" assignment for stations for which SAS JPM analysis has NOT been performed, but are in the same geographical area as the stations examined in the first pass. These are the stations for which the Access Flag is lowered, and the "pending" assignment has been calculated from the "current" (historical) assignment.

In either case, the user's job will be to "accept" the "pending" assignment, whether it's the one placed there by the administrator during the reset, or updated in the table by the user based on the
assignments made on the surrounding stations. The "accept" function will be implemented as a button on the view interface, and will cause the symbology of the station to be encircled with a black outline (see below).

PTC Station Symbology

We discussed various approaches to symbolizing the PTC station locations. The problem we're facing is that we need to symbolize the stations on a combination of values and fields: Does the station have a "pending" assignment of 1, 2, 3, 4, 5, 6, or 7; is the origin of the "pending" assignment a SAS analysis or a "current" (historical) assignment; and has the site been "accepted" or not?

We're going to deal with the SAS analysis vs. "historical" assignment problem by calc'ing the "historical" assignments (1 through 7) to be their arithmetic inverse -- that is, a SAS assignment of 6 will be distinguishable from a "historical" assignment of -6. This will be handled at the once-a-year system reset by the application administrator. These will be stored in the "pending" field, using
bright "neon" colors to represent positive values and matching pastel colors to represent the negative values.

The other problem is representing the "accepted" flag. To do this we will create a second theme from the same data source, and have it draw on top of the theme described above; an "unaccepted" PTC site will not be drawn in this theme, but an "accepted" one will draw as a black-outlined shape matching the symbols used in the "pending" theme. This way the user will be able to tell at a glance if the site is accepted or unaccepted, the source of the "pending" assignment, and the assignment itself. We think this will work.

Yearly "reset" message

When the project opens, after a global variable is checked to see whether or not to display a "nag" message regarding system reset time (see below), an Access connection should be established. Then, the user should be prompted with a listbox to select the county or counties in which s/he wishes to work. There should also be a 'box' option to allow a box to be used to specify the study area. Whichever method is chosen, it should be used to do an up-front filter on both PTC themes described above (to prevent all 56,000 of them from drawing every time). The mapextent should be set to the specified area, and the locator map should be updated.

A "nag message" should be displayed at startup, before the DB connect, by checking the value of a project global variable set by the administrator, when the program is within 30 days of the "reset" date. Between 30 and 0 days of that date, the nag screen should be dismissable and the user should be able to work. After the reset date, the user should be locked out of the application (or at least the DB connect) until the Administrator "resets" the PTC file. Most of this work will be handled in Access, but we will provide a small administrator-level script to reset the global variable.

The administrator will perform a reset once a year, on or shortly after October 1st. The reset will notify access to update the "current assignment" from the "pending assignment" for "accepted" PTCs; to update the associated date stamp for "accepted" PTCs, to lower all the "accepted" and "access" flags, and then to raise "access" flags and populate the best-fit and "pending" fields for the next year's set of PTCs.

Attachment F - Initial Program Setup Procedures

The ArcView program is set to run from a Microsoft Windows platform. Once the computer's ODBC parameters are set, the program can be opened and manipulated. The initial setup requires the administrator's password to save the program once changes have been made.

1. Make sure the system DSN ODBC driver, named 'SEASCOV', is installed. If it is not, create the driver by copying the MS-Access driver version 3.50.360200 (or later), and naming it 'SEASCOV'. This can be achieved using the Windows ODBC Data Source Administrator (click Start > Control Panel > 32bitODBC icon). Configure the driver to point to the Access database used as the import table source, *Table1* in ArcView.

2. Upload the project directory onto the C:/ drive. This will give the directory the path C:/seasonal/*. If a different drive or path is used, then the project ASCII file (*beta.apr*) will need to be changed. This can be done from any text editor with the find...replace options.

3. Open the program. If the ODBC connection is bad, an error message will display warning the user that *Table1* does not exist. If this occurs, shut the program down and reinstall/configure the ODBC driver.

4. Once the project opens correctly, the PTC theme in the table-of-contents will need to be classified. A legend (.avl) file already exists for the theme. To load the file, double-click on the theme in the table-of-contents and click the 'Load...' button. Choose *ptc.avl* from the Load Legend dialog box.

5. Save the project (remember an administrator's password is required), close the program, and reopen to check for errors.

SUPPLEMENT A:

AVENUE PROGRAM SCRIPTS

```
'''
''' Date.Warning
'''
''' smm 05-01-99
'''
''' Checks the current date and matches against the RESET date.  If
''' the current date is past the reset date, the user will be prompted
''' for a password in order to continue.
''' The user is also issued a messagebox with the current date and a
''' countdown until the reset once within 30 days of the reset date.
'''
''' This script is run from the startup script MyProject.StartUp during
''' the project startup.
'''
''' Notes:  The reset date will change according to when it is scheduled
'''         and rescheduled on an anual basis.
''' ---------------------------------------------------------------------

''' Setting the date format and capturing the date as a string.
 Date.SetDefFormat("yyyyMMdd")
 today = Date.Now
 aDate = today.AsString

'''THE RESET DATE (THIS WILL CHANGE ACCORDING TO THE TRUE RESET DATE)
 theDeadLine = "19991231"

''' If the current date is within 30 days, give the user a message with
''' the current date and another with a countdown of how many days are
''' left until the reset date.  If the current date is one day prior
''' to the reset, then issue a special message box saying so.
 theDaysLeft = (theDeadLine.AsNumber - aDate.AsNumber)
 if (theDaysLeft <= 30) then
   MsgBox.Info("today is"++today.AsString,"")
   if (theDaysLeft = 1) then
     MsgBox.Info("There is 1 day left before the reset","")
   else
     MsgBox.Info("There are"++theDaysLeft.AsString++"days left before reset","")
   end
 end

'''If the current date is on or past the reset date, prompt the user
'''for a password.  If the password is incorrect, bail out.
 if (theDaysLeft <= 0) then
   MsgBox.Info("The reset day was scheduled for"++theDeadLine++".","")
   aPass = MsgBox.Password
   if (aPass = _thePass) then
     MsgBox.Info("You're in!!!","")
   else
     MsgBox.Info("Your password was not correct.  This project will shut down.","")
     av.Run("Project.Exit",nil)
   end
 end
```

```
'''
''' Dialog Editor.SaveDetached
'''
''' smm 05-01-99
'''
''' Script is run by choosing Save Detached... from the File menu
''' on the project window interface.
'''
''' Customized system script associated with the Save Detached...
''' menu option from the File menu in the Project window interface.
''' In this script, the user is prompted for a password before
''' saving as a detached file.
'''
''' Notes:  Save Detached detaches the project from the Dialog
'''         Designer. This option creates a new project that
'''         contains all your dialogs but without the dialog
'''         editor.  There is a possibility of inadvertently saving
'''         the detached project with the same name as this project,
'''         thus overwriting the original.  For this reason, the
'''         script acts as a safety measure to only allow those
'''         with a password to 'Save Detached...'
'''
'''         The password global variable _thePass is set in the
'''         script MyProject.StartUp
''' ----------------------------------------------------------------

theProject = av.GetProject

''' Prompt for a password.  If correct, continue with Save Detached...

aPass = MsgBox.Password
if (aPass = _thePass) then

''' From this point until the 'else' statement of the current
''' 'If...then' loop, the script is a copy of the ArcView system
''' script 'Dialog Editor.SaveDetached'.

''' Save changes to current project before creating the new one
 if (theProject.IsModified) then
  res = MsgBox.YesNo("Changes will be saved to "+ theProject.GetName
                + " before the new detached project is written. Do you want to continue?",
                "Save Detached", TRUE)
  if (res = FALSE) then return nil end
  if (res) then
   av.Run("Project.Save", nil)
   if (theProject.IsModified) then return nil end
  end
 end

''' Load the dialog core extension

 coreExt = Extension.Open("$AVEXT/dlogcore.___".AsFilename)
 if (coreExt = nil) then
  msgbox.Error("Can't unload the Dialog Designer because $AVEXT/avdlogcore.___ failed to load.","Error")
  return nil
 end

''' Get new filename

 if ((System.GetEnvVar("HOME") <> nil) and File.IsWritable("$HOME".AsFileName)) then
  defName = FileName.Make("$HOME").MakeTmp("proj", "apr")
 else
  defName = FileName.Make("proj1.apr")
 end

 theFName = FileDialog.Put(defName, "*.apr", "Save Detached")
 if (theFName = nil) then return nil end
 theProject.SetFileName(theFName)
```

```
''' Embed the dialogs

 ddext = Extension.GetExtensions.Get("Dialog Designer")
 dedGUI = Extension.FindGui("DialogEditor")
 theDialogEditors = theProject.GetDocsWithGUI(dedGUI)

 for each d in theDialogEditors
  ''' See if a dialog with the same name is already in the project
  if ( av.GetProject.GetDialogs.Get(d.GetName) <> nil) then
   replace = MsgBox.YesNo( "There is already a dialog called"++d.GetName+" in the project."
                  +"Do you want to overwrite it?",
                   "Embed Dialog", TRUE)
   if (replace.Not) then
    msgbox.Error("Save Detached failed because there is more than one dialog with the same name.","Error")
    coreExt.Unload
    return nil
   end
  end

  theProject.AddDialog(d.GetDialog)
  theProject.RemoveDoc(d)
 end

''' Put all graphic controls on views and layouts in run mode

 docList = theProject.GetDocs
 for each d in docList
  if ((d.Is(View)) or (d.Is(Layout) and d.Is(DialogEditor).Not)) then
   graList = d.GetGraphics
   gcList = graList.FindAllByClass(GraphicControl)
   if (gcList <> nil) then
    for each gc in gcList
     gc.SetEditable(False)
    end
   end
  end
 end

 PropWin.The.Close

 if (av.FindDialog("Control Tools").IsOpen) then
  av.FindDialog("Control Tools").Close
 end

''' Set the Dialog Designer CanUnload script to nil
 ddext.SetCanunloadScript(nil)

''' Unload the Dialog Designer extension
 while (ddext <> nil)
  if (ddext.Unload) then
   ddext = Extension.GetExtensions.Get("Dialog Designer")
  else
   msgbox.Error("Can't unload the Dialog Designer","Error")
   coreExt.Unload
   break
  end
 end

''' Save the new detached project
 if (theProject.Save) then
  av.ShowMsg( "Detached Project saved to '"+theProject.GetFileName.GetBaseName+"'")
  if (System.GetOS = #SYSTEM_OS_MAC) then
   realFName = theProject.GetFileName
   if (nil <> realFName) then
    Script.Make("MacClass.SetDocInfo(SELF, Project)").DoIt(realFName)
   end
  end
 end
```

```
''' This ends the system script copy.

''' If the password is incorrect, give the user a message box and
''' bail from routine.

else
  MsgBox.Error("Your password is incorrect!  This project will not be saved.","")
end
```

```
'''
''' Image.BestFitCurves
'''
''' smm 05-01-99
'''
''' Run when the ATR Graphs button on the View window is clicked.
'''
''' This script creates an image window of all seven atr group graphs.
''' It's associated with the CLICK field of the ATR Graphs button on the
''' View window.
'''
''' Files Called:  allgroup.gif
'''
''' Notes:  The image file is hardcoded and will need to be changed if
'''         the project is relocated to a different server with
'''         a different pathname.  See 'setup.txt'
'''
'''         The original size and position of the image is in pixel
'''         units.  They were originally set to center on a 21" screen
'''         with a desktop area set to 1024 by 768 pixels.
'''         These numbers can be adjusted to fit other screens.
''' ----------------------------------------------------------------------

theImage = "c:\seasonal\images\group_images\allgroup.gif".AsFileName
anImageWin = ImageWin.Make(theImage, "ATR Graphs")
anImageWin.Open

anImageWin.MoveTo(150,75)
anImageWin.Resize(700,600)
```

```
'''
''' Image.Group1
'''
''' smm 05-01-99
'''
''' Run when Group1 is chosen from the ATR Groups menu in the View
''' window interface.
'''
''' Image.Group1 launches an image window with the atr graph for
''' Group 1.
'''
''' Files Called:  group1.gif
'''
''' Notes:  The image file is hardcoded and will need to be changed if
'''         the project is relocated to a different server with
'''         a different pathname.  See 'setup.txt'
'''
'''         The original size and position of the image is in pixel
'''         units.  They were originally set to center on a 21" screen
'''         with a desktop area set to 1024 by 768 pixels.
'''         These numbers can be adjusted to fit other screens.
''' ----------------------------------------------------------------

theImage = "c:\seasonal\images\group_images\group1.gif".AsFileName
anImageWin = ImageWin.Make(theImage, "Group 1 Graph")
anImageWin.SetScaled(false)
anImageWin.Open
anImageWin.Resize(550, 500)
anImageWin.MoveTo (10, 150)
```

```
'''
''' Image.Group2
'''
''' smm 05-01-99
'''
''' Run when Group2 is chosen from the ATR Groups menu in the View
''' window interface.
'''
''' Image.Group2 launches an image window with the atr graph for
''' Group 2.
'''
''' Files Called:  group2.gif
'''
''' Notes:  The image file is hardcoded and will need to be changed if
'''         the project is relocated to a different server with
'''         a different pathname.  See 'setup.txt'
'''
'''         The original size and position of the image is in pixel
'''         units.  They were originally set to center on a 21" screen
'''         with a desktop area set to 1024 by 768 pixels.
'''         These numbers can be adjusted to fit other screens.
''' ---------------------------------------------------------------------

theImage = "d:\seasonal\images\images_groups\group2.gif".AsFileName
anImageWin = ImageWin.Make(theImage, "Group 2 Graph")
anImageWin.SetScaled(false)
anImageWin.Open
anImageWin.Resize(550, 500)
anImageWin.MoveTo (10, 150)
```

```
'''
''' Image.Group3
'''
''' smm 05-01-99
'''
''' Run when Group3 is chosen from the ATR Groups menu in the View
''' window interface.
'''
''' Image.Group3 launches an image window with the atr graph for
''' Group 3.
'''
''' Files Called:  group3.gif
'''
''' Notes:  The image file is hardcoded and will need to be changed if
'''         the project is relocated to a different server with
'''         a different pathname.  See 'setup.txt'
'''
'''         The original size and position of the image is in pixel
'''         units.  They were originally set to center on a 21" screen
'''         with a desktop area set to 1024 by 768 pixels.
'''         These numbers can be adjusted to fit other screens.
''' ----------------------------------------------------------------

theImage = "d:\seasonal\images\images_groups\group3.gif".AsFileName
anImageWin = ImageWin.Make(theImage, "Group 3 Graph")
anImageWin.SetScaled(false)
anImageWin.Open
anImageWin.Resize(550, 500)
anImageWin.MoveTo (10, 150)
```

```
'''
''' Image.Group4
'''
''' smm 05-01-99
'''
''' Run when Group4 is chosen from the ATR Groups menu in the View
''' window interface.
'''
''' Image.Group4 launches an image window with the atr graph for
''' Group 4.
'''
''' Files Called:  group4.gif
'''
''' Notes:  The image file is hardcoded and will need to be changed if
'''         the project is relocated to a different server with
'''         a different pathname.  See 'setup.txt'
'''
'''         The original size and position of the image is in pixel
'''         units.  They were originally set to center on a 21" screen
'''         with a desktop area set to 1024 by 768 pixels.
'''         These numbers can be adjusted to fit other screens.
''' ---------------------------------------------------------------------

theImage = "d:\seasonal\images\images_groups\group4.gif".AsFileName
anImageWin = ImageWin.Make(theImage, "Group 4 Graph")
anImageWin.SetScaled(false)
anImageWin.Open
anImageWin.Resize(550, 500)
anImageWin.MoveTo (10, 150)
```

```
'''
''' Image.Group5
'''
''' smm 05-01-99
'''
''' Run when Group5 is chosen from the ATR Groups menu in the View
''' window interface.
'''
''' Image.Group5 launches an image window with the atr graph for
''' Group 5.
'''
''' Files Called: group5.gif
'''
''' Notes:  The image file is hardcoded and will need to be changed if
'''         the project is relocated to a different server with
'''         a different pathname.  See 'setup.txt'
'''
'''         The original size and position of the image is in pixel
'''         units.  They were originally set to center on a 21" screen
'''         with a desktop area set to 1024 by 768 pixels.
'''         These numbers can be adjusted to fit other screens.
''' ----------------------------------------------------------------

theImage = "d:\seasonal\images\images_groups\group5.gif".AsFileName
anImageWin = ImageWin.Make(theImage, "Group 5 Graph")
anImageWin.SetScaled(false)
anImageWin.Open
anImageWin.Resize(550, 500)
anImageWin.MoveTo (10, 150)
```

```
'''
''' Image.Group6
'''
''' smm 05-01-99
'''
''' Run when Group6 is chosen from the ATR Groups menu in the View
''' window interface.
'''
''' Image.Group6 launches an image window with the atr graph for
''' Group 6.
'''
''' Files Called: group6.gif
'''
''' Notes:  The image file is hardcoded and will need to be changed if
'''         the project is relocated to a different server with
'''         a different pathname.  See 'setup.txt'
'''
'''         The original size and position of the image is in pixel
'''         units.  They were originally set to center on a 21" screen
'''         with a desktop area set to 1024 by 768 pixels.
'''         These numbers can be adjusted to fit other screens.
''' ---------------------------------------------------------------------

theImage = "d:\seasonal\images\images_groups\group6.gif".AsFileName
anImageWin = ImageWin.Make(theImage, "Group 6 Graph")
anImageWin.SetScaled(false)
anImageWin.Open
anImageWin.Resize(550, 500)
anImageWin.MoveTo (10, 150)
```

```
'''
''' Image.Group7
'''
''' smm 05-01-99
'''
''' Run when Group7 is chosen from the ATR Groups menu in the View
''' window interface.
'''
''' Image.Group7 launches an image window with the atr graph for
''' Group 7.
'''
''' Files Called: group7.gif
''' Notes:  The image file is hardcoded and will need to be changed if
'''        the project is relocated to a different server with
'''        a different pathname.  See 'setup.txt'
'''
'''        The original size and position of the image is in pixel
'''        units.  They were originally set to center on a 21" screen
'''        with a desktop area set to 1024 by 768 pixels.
'''        These numbers can be adjusted to fit other screens.
''' ------------------------------------------------------------------

theImage = "d:\seasonal\images\images_groups\group7.gif".AsFileName
anImageWin = ImageWin.Make(theImage, "Group 7 Graph")
anImageWin.SetScaled(false)
anImageWin.Open
anImageWin.Resize(550, 500)
anImageWin.MoveTo (10, 150)
```

```
'''
''' MyDialog.Activate
'''
''' smm 04-99
'''
''' This script is a place holder for
''' the PGA_Update dialog's Activate
''' property.  The script does nothing but
''' prevent an error occuring when the
''' dialog box becomes active.

''' If I can find a way to delete this
''' script from the Activate field of the
''' PGA Update Dialog Box, this script
''' will be deleted
```

```
'''
''' MyDialog.Bt_Hotlink.Click
'''
''' smm 06-99
'''
''' This script is inactive in the current project.  It was intended
''' to be associated with a button inside the dialog box that would
''' launch an image window with the SAS image report of the station
''' that is selected inside the listbox.

''' However, for reasons not understood, an image window can not
''' be launched from the dialog box.  A reason and solution
''' for this error is under investigation
''' ----------------------------------------------------------------

theCovSysID=_aListBox.GetSelection
theVal = (theCovSysID.Get(0))
theVal = "d:\seasonal\htimages\station"+theVal+".gif"
if (File.Exists(theVal.AsFileName)) then
  i = ImageWin.Make(theVal.AsFileName, theVal)
  i.SetScaled(false)
  i.Open
  i.Resize(650,700)
  i.MoveTo(375,50)
else
  MsgBox.Warning("File "+theVal+" not found.","Hot Link")
end
```

```
'''
''' MyDialog.Close
'''
''' smm 06-99
'''
''' Runs when the PGA_Update dialog box closes (is associated with the
''' dialog box's CLOSE field).
'''
''' Scripts called:  MyProject.RemoveLabels
'''
''' This script runs MyProject.RemoveLabels to remove the CovSysID
''' labels that are generated during the opening of the dialog box.
'''
''' The script also deletes the temp file newfile each time
''' the dialog box is closed or the Accept PGA button
''' is clicked.
''' ----------------------------------------------------------------------

av.Run("MyProject.RemoveLabels",nil)

theProject = av.GetProject
aTableDelete = theProject.FindDoc("newfile")
av.GetProject.RemoveDoc( aTableDelete )
```

```
'''
''' MyDialog.ExportVTab.FromTable
'''
''' smm 04-99
'''
''' Runs when the 'Open PGA Update Dialog Box' button on the Table window's button bar
''' is clicked (associated with the button's CLICK field).
'''
''' This script creates the 'newfile' table and opens the PGA Update
''' Dialog Box. The dialog box's listbox is populated with the
''' contents of newfile when the dialog box opens (refer to the dialog's
''' open script MyDialog.Open). Newfile is the temporary table that is written
''' to while edits are being made in the dialog box. Once applied, the edits
''' are written from 'newfile' to the Access table via SQL.
'''
''' Notes: Opens the dialog PGA Update Dialog Box
'''
''' ------------------------------------------------------------------------------

''' Check to see if Attributes of PTC Stations is active. If not, send
''' the user an error message.
 theTable = av.GetActiveDoc
 theTableCheck = av.GetActiveDoc.AsString
 aStr = "Attributes of PTC Stations"
 if ((theTableCheck = aStr).Not) then
   MsgBox.Error("You must make the Attributes of PTC Stations table active!","")

''' If the correct table is active, then continue...
 else
   theVTab = theTable.GetVTab
   ''' create a new dBASE file of the table's selected records
   theVTab.Export ( "newfile".asFileName, dBASE, TRUE )
   ''' output file, newfile.dbf, contains only the selected
   ''' records

'''Now create a VTab from newfile.dbf and set the name to "newfile"
   theFileName = "newfile.dbf".asFileName
   theVTab = VTab.Make (theFileName, false, false)
   theTable = Table.Make(theVTab)
   theName = theTable.SetName ("newfile")

'''Finally, find and open the dialog box "PGA_Update". At this point,
'''MyDialog.Open runs.
   theDialog = av.GetProject.FindDialog("PGA_Update")
   theDialog.Open
 end
```

```
'''
''' MyDialog.ExportVTab.FromView
'''
''' smm 04-99
'''
''' Runs when the 'Open PGA Update Dialog Box' button on the View window's button bar
''' is clicked (associated with the button's CLICK field).
'''
''' This script creates the 'newfile' table and opens the PGA Update
''' Dialog Box.  The dialog box's listbox is populated with the
''' contents of newfile when the dialog box opens (refer to the dialog's
''' open script MyDialog.Open).  Newfile is the temporary table that is written
''' to while edits are being made in the dialog box.  Once applied, the edits
''' are written from 'newfile' to the Access table via SQL.
'''
''' Notes:  Opens the dialog PGA Update Dialog Box
'''
''' ------------------------------------------------------------------------------

''' Check to see if Attributes of PTC Stations is active.  If not, send
''' the user an error message.
 theTable = av.GetProject.FindDoc("Attributes of PTC Stations")
 aView = av.GetProject.FindDoc("NCSU-ITRE Seasonal Group Assignment Application")
 theThemeCheck = aView.GetActiveThemes
 aStr = "PTC Stations"
   for each i in theThemeCheck
 if ((i.AsString = aStr).Not) then
   MsgBox.Error("You must make PTC Stations an active theme!","")

'''if the correct theme is active, continue...
 else
   theVTab = theTable.GetVTab
   ''' create a new dBASE file of the table's selected records
   theVTab.Export ( "newfile".asFileName, dBASE, TRUE )
   ''' output file, newfile.dbf, contains only the selected records

'''Now create a VTab from newfile.dbf and set the name to "newfile"
   theFileName = "newfile.dbf".asFileName
   theVTab = VTab.Make (theFileName, false, false)
   theTable = Table.Make(theVTab)
   'theTable.GetWin.Open
   theName = theTable.SetName ("newfile")

'''Finally, find and open the dialog box "PGA_Update".  At this
'''MyDialog.Open runs.
   theDialog = av.GetProject.FindDialog("PGA_Update")
   theDialog.Open
 end
 end
```

```
'''
''' MyDialog.Lbt_Accept.Click
'''
''' smm 04-99
'''
''' Runs when the PGA Update Dialog Box's ACCEPT button is clicked.
'''
''' Calculates a value reflecting the current date for the AcceptFlag field
''' in the table 'newfile' for the selected records in the PGA Update db's
''' listbox.
'''
''' Scripts Called:  MyDialog.Lbx_Newfile.Update by broadcast
'''                  MyDialog.Lbt_UnAccept.Update by broadcast
'''
''' Notes:  Broadcasts to run MyDialog.Lbx_Newfile.Update to recreate the
'''         listbox after newfile is refreshed, and MyDialog.Lbt_UnAccept.Update
'''         to make the UnAccept button active.
'''
''' ------------------------------------------------------------------------

''' Capture 'newfile' and set it editable, and capture its AcceptFlag field.
 aVtab =  av.GetProject.FindDoc("newfile").GetVTab
 aVTab.SetEditable(True)
 aField = aVTab.FindField("AcceptFlag")

'''Capture the current date and return it as a string
 today = Date.Now
 today.SetFormat( "yyyyMMdd" )
 aStr = today.AsString

'''Calculate the current date in AcceptFlag as a number
 aVTab.Calculate(aStr, aField)

'''Refresh the newfile table and turn editing mode off
 aVTab.Refresh
 aVTab.SetEditable(False)

'''Send a broadcast to to rebuild the listbox with the refreshed newfile (by
'''running the listbox update script MyDialog.Lbx_Newfile.Update) and
'''make the UNACCEPT button active.
 self.BroadcastUpdate

'''Disengage the Accept button so that it can not be used while the
'''UnAccept button is enabled (this creates a toggle
'''effect between Accept and UnAccept).
 self.SetEnabled(false)
```

```
""
"" MyDialog.GroupButtons.Update
""
"" smm 05-99
""
"" Runs when an broadcast is sent from the following scripts:
""
""        MyDialog.Lbx_Newfile.Update
""
"" Enables the Update buttons in the dialog box.
"" This script is associated with the Update script
"" field for each of the seven update buttons (labeled
"" one thru seven).
""
"" ------------------------------------------------------------

self.SetEnabled(true)
```

```
'''
''' MyDialog.Lbt_Accept.Update
'''
''' smm 04-99
'''
''' Runs when a broadcast is sent from:
'''
'''         MyDialog.Lbt_UnAccept.Click
'''         MyDialog.Lbx_Newfile.Select
'''
''' Enables the ACCEPT button in the PGA Update Dialog Box.  The script
''' is associated with the ACCEPT button's Update property.  The button
''' will only be enabled if the AcceptFlag value in newfile for the
''' selected record in the listbox 'Lbx_Newfile' has a value of '0'.
'''
''' Notes:  A broadcast is sent from the above scripts to both the ACCEPT
'''         and UNACCEPT buttons at the same time.  If the AcceptFlag value
'''         is anything other than '0', then it HAS been accepted and the
'''         ACCEPT button will not be enabled.  However, the UNACCEPT button
'''         will enable and become
'''         visible.  Since the ACCEPT and UNACCEPT buttons are located on
'''         top of one-another inside the dialog box, the result of the
'''         constraints is a toggle-effect between the two buttons (refer to
'''         MyDialog.Lbt_UnAccept.Update).
'''
''' ------------------------------------------------------------------------

''' If the AcceptFlag value is '0', then enable the ACCEPT button.
aVTab = av.GetProject.FindDoc("newfile").GetVTab
aField = aVTab.FindField("AcceptFlag")
for each rec in aVTab.GetSelection
 aVal = aVTab.ReturnValue(aField, rec)
 if (aVal = 0) then
  self.SetEnabled(true)
  self.SetVisible(true)

''' If it is anything other than '0', set the button's enabled and visible
''' properties to false.
 else
  self.SetEnabled(false)
  self.SetVisible(false)
 end
end
```

```
'''
''' MyDialog.Lbt_Apply.Click
'''
''' smm 04-99
'''
''' Runs when the PGA Update Dialog Box's APPLY button is clicked (associated
''' with the APPLY button's CLICK property)
'''
''' When clicked, the Apply button calls this script which queries the
''' Access table and writes changes to it and QTABLE from the temporary file 'newfile'.
''' The unique ID for each station held in the attribute CovSysID is
''' captured for each record in newfile. This value is queried against the
''' unique identifier in the Access table (Table1) during the rewrites via SQL. The
''' writes are also made to QTABLE, which is created at the project's startup from
''' a user-defined selected set (by County) from the Access database. QTABLE in
''' turn is joined to Attributes of PTC Stations.
'''
''' There are three attributes in Table1 and QTABLE that are written to:
'''
'''     PGA (Pending Group Assignment for each station)
'''     AcceptFlg (Date stamp for recording the date of change for the PGA)
'''     Classify (Used to classify the PTC Stations theme in ArcView)
'''
''' Once the edits are made, Attributes of PTC Stations (which is joined to
''' QTABLE) is refreshed to show changes.
'''
''' Notes: When refreshing a table in ArcView, any joins that are established are rejoined,
'''       and the table is requeried if it is established via an SQL connection. Such
'''       processes cost heavily in time. For this reason, Table1 is not refreshed after
'''       writes to it are made. Only "Attributes of PTC Stations" is refreshed to
'''       indicate changes that have taken place. For this reason, IF TABLE1 IS OPENED
'''       DURING THE DURATION OF THE EDITING SESSION IN ARCVIEW, THE CHANGES WILL NOT APPEAR
'''       TO HAVE TAKEN PLACE. Table1 can be refreshed using the Refresh option from
'''       the Table's Table menu option to see the changes if desired. When the project
'''       is reopened, the edits that were made in the previous session will appear.
'''       This is because Table1 is requeried at project startup (see Query.Table1).
'''
''' ----------------------------------------------------------------------------
'''
''' Find newfile and Table1, and establish connection with the ODBC connection
''' set to the global variable "_seasonal" in MyProject.StartUp
aVTab =  av.GetProject.FindDoc("newfile").GetVTab
aTable = av.GetProject.FindDoc("Table1")
theSQLCon = SQLCon.Find(_seasonal)
theSQLCon.Login("")

'''Now we loop thru newfile to write edits to Table1 in Access.

'''First, set variables for the join field covsysid and any fields that will
'''be written to.
for each rec in aVTab
  aRelateField = aVTab.FindField("CovSysID")
  aJoinVal = aVTab.ReturnValueString(aRelateField,rec)
  aPGAField = aVTab.FindField("PGA")
  aPGAVal = aVTab.ReturnValueString(aPGAField,rec)
  aPGAValNum = aVTab.ReturnValue(aPGAField,rec)
  anAcceptField = aVTab.FindField("AcceptFlag")
  anAcceptVal = aVTab.ReturnValueString(anAcceptField,rec)
  anAccessFlg = aVTab.FindField("AccessFlag")
  anAccessVal = aVTab.ReturnValue(anAccessFlg,rec)
  anExpression = "(((tblSeasonalFactorStations.CovSysID) =" ++aJoinVal++ "))"

''' Now for the SQL's:
'''     The first writes the new PGA value to Access via a match between
'''     the newfile and the Access tables' CovSysID values (using with the
'''     expression variable anExpression above)
  UpdatePGA = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.PGA =" ++aPGAVal++ "WHERE" +↵
  theSQLCon.ExecuteSQL(UpdatePGA)
```

```
''' Next we write the current date to the AcceptFlg field in the Access table.
  UpdateAcceptFlg = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.AcceptFlag =" ++anAcceptVal+
  theSQLCon.ExecuteSQL(UpdateAcceptFlg)

''' And finally, we'll rewrite the Classify field.

''' The original values of Classify is the same as that of the PGA.  However, the number
''' is multiplied by either -1, 100, or both if certain situations occur:
'''
'''     If the AccessFlag is '0' then the value will be multiplied by -1,
'''     If the AccessFlag is greater than '0', the number will remain positive,
'''     If the AcceptFlag is greater than '0', i.e. the station has been 'accepted', then
'''     the Classify value (negative or positive) is multiplied by 100.
'''
''' This gives four possible scenerios of which we'll classify the PTC Stations theme by:
'''   1.  Station has data but is not accepted (1-7)
'''   2.  Station has data and is accepted (100-700)
'''   3.  Station has no data and is not accepted ([-1]-[-7])
'''   4.  Station has no data and is accepted ([-100]-[-700])

  if (anAccessVal = "0") then
    if (anAcceptVal = "0") then
      aNegPGAVal = (aPGAValNum * -1).AsString
      UpdateClass = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.Classify =" ++aNegPGAVal++ "V
      theSQLCon.ExecuteSQL(UpdateClass)
    else
      aNegPGAVal = (aPGAValNum * -100).AsString
      UpdateClass = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.Classify =" ++aNegPGAVal++ "V
      theSQLCon.ExecuteSQL(UpdateClass)
    end
  else
    if (anAcceptVal = "0") then
      UpdateClass = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.Classify =" ++aPGAVal++ "WHE
      theSQLCon.ExecuteSQL(UpdateClass)
    else
      aPGAVal = (aPGAValNum * 100).AsString
      UpdateClass = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.Classify =" ++aPGAVal++ "WHE
      theSQLCon.ExecuteSQL(UpdateClass)
    end
  end
end

''' Now we'll write the edits to QTABLE1 so when Attributes of PTC Stations is refreshed, the
''' changes will be reflected in the View.
aVTab =  av.GetProject.FindDoc("newfile").GetVTab
aTable = av.GetProject.FindDoc("qtable").GetVTab

for each rec in aVTab
  aRelateField = aVTab.FindField("CovSysID")
  aJoinVal = aVTab.ReturnValueString(aRelateField,rec)
  aPGAField = aVTab.FindField("PGA")
  aPGAVal = aVTab.ReturnValueString(aPGAField,rec)
  aPGAValNum = aVTab.ReturnValue(aPGAField,rec)
  anAcceptField = aVTab.FindField("AcceptFlag")
  anAcceptVal = aVTab.ReturnValueString(anAcceptField,rec)
  anAccessFlg = aVTab.FindField("AccessFlag")
  anAccessVal = aVTab.ReturnValue(anAccessFlg,rec)

  aTable.SetEditable(True)
  aBitmap = aTable.GetSelection
  theExpression = "([CovSysID] = "+aJoinVal+")"
  aTable.Query( theExpression, aBitmap, #VTAB_SELTYPE_NEW)
  aTable.UpdateSelection

  aField = aTable.FindField("PGA")
  aTable.Calculate(aPGAVal, aField)

  aField = aTable.FindField("AcceptFlag")
  aTable.Calculate(anAcceptVal, aField)
```

```
  aField = aTable.FindField("Classify")
  if (anAccessVal = "0") then
    if (anAcceptVal = "0") then
      aNegPGAVal = (aPGAValNum * -1).AsString
      aTable.Calculate(aNegPGAVal, aField)
    else
      aNegPGAVal = (aPGAValNum * -100).AsString
      aTable.Calculate(aNegPGAVal, aField)
    end
  else
    if (anAcceptVal = "0") then
      aTable.Calculate(aPGAVal, aField)
    else
      aPGAVal = (aPGAValNum * 100).AsString
      aTable.Calculate(aPGAVal, aField)
    end
  end
end

aTable.SetEditable(False)

"' PTC Stations table must be refreshed to show changes in the View and
"' the dialog box when newfile is created during editing.
theTable = av.getproject.finddoc("Attributes of PTC Stations")
theVTab = theTable.GetVTab
theVTab.Refresh

"' And finally close the PGA Update Dialog Box once the edits are made.
self.GetDialog.Close
```

```
'''
''' MyDialog.Lbt_Apply.Update
'''
''' smm 4-99
'''
''' Runs when a broadcast is sent from:
'''
'''         MyDialog.Lbx_Newfile.Select
'''
''' This script enables the Apply button in the PGA Update
''' Dialog Box.  It is associated with the APPLY button's
''' UPDATE property.
'''
''' ---------------------------------------------------

self.SetEnabled(true)
```

```
'''
''' MyDialog.Lbt_Cancel.Click
'''
''' smm 04-99
'''
''' Runs whenever the PGA Update Dialog Box's
''' CANCEL button is clicked (associated
''' with the CANCEL button's CLICK field).
'''
''' Dismisses the dialog box without writing any edits
''' to the QTABLE or Table1.
'''
''' Notes: When the Close request is sent to the PGA Update dialog,
'''       its CLOSE script (MyDialog.Close) executes.
'''
''' ------------------------------------------------------------------

self.GetDialog.Close
```

```
'''
''' MyDialog.Lbt_UnAccept.Click
'''
''' smm 04-99
'''
''' Runs when the PGA Update Dialog Box's UNACCEPT button is
''' clicked (associated with the UNACCEPT button's CLICK property).
'''
''' Scripts called:
'''
'''         MyDialog.Lbx_Newfile.Update by broadcast
'''         MyDialog.Lbt_Accept.Update
'''
''' Calculates a '0' for the listbox's selected records'
''' AcceptFlag field.  This value is a calculated into the
''' temporary file 'newfile'.
'''
''' ----------------------------------------------------------------

aVtab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("AcceptFlag")
aStr = "0"
aVTab.Calculate(aStr, aField)

''' Refresh newfile to reflect changes
aVTab.Refresh
aVTab.SetEditable(False)

''' To toggle back to the Accept button, first set
''' the UnAccept button's enable feature to false:
self.SetEnabled(false)

''' Now send a broadcast to run the Accept button's UPDATE script
''' MyDialog.Lbt_Accept.Update to enable the Accept button.
''' A broadcast is also sent to the listbox to run
''' MyDialog.Lbx_Newfile.Update to rebuild the listbox with
''' the refreshed newfile values.
self.BroadcastUpdate
```

```
'''
''' MyDialog.Lbt_UnAccept.Update
'''
''' smm 04-99
'''
''' Executes when a broadcast is sent from either of the following scripts:
'''
'''         MyDialog.Lbt_Accept.Click
'''         MyDialog.Lbx_Newfile.Select
'''
''' Enables the Update PGA Dialog Box's UNACCEPT button when a broadcast is
''' sent from either of the two script above AND the AcceptFlag value for
''' the selected record in the dialog box's listbox has a value greater than
''' '0'.
'''
''' Notes:  A broadcast is sent from the above scripts to both the ACCEPT
'''         and UNACCEPT buttons at the same time.  If the AcceptFlag value
'''         is '0' then it has not been accepted and the UNACCEPT button will
'''         not be enabled.  However, the ACCEPT button will enable and become
'''         visible.  Since the ACCEPT and UNACCEPT buttons are located on
'''         top of one-another inside the dialog box, the result of the
'''         constraints is a toggle-effect between the two buttons (refer to
'''         MyDialog.Lbt_Accept.Update).
''' ----------------------------------------------------------------------

''' Enable the UNACCEPT button if the AcceptFlag value for the selected
''' record in the listbox is anything other than '0'.
aVTab = av.GetProject.FindDoc("newfile").GetVTab
aField = aVTab.FindField("AcceptFlag")
for each rec in aVTab.GetSelection
 aVal = aVTab.ReturnValue(aField, rec).AsString
 if (aVal <> 0) then
  self.SetEnabled(true)
  self.SetVisible(true)

''' Else, set enabling and visible properties to false.
 else
  self.SetEnabled(false)
  self.SetVisible(false)
 end
end
```

```
'''
''' MyDialog.Lbx_Newfile.Apply
'''
''' smm 04-99
'''
''' Executed when a record in the Lbx_Newfile listbox (on the
''' Update PGA Dialog Box) is double-clicked.  It is associated with the
''' listbox's APPLY field.
'''
''' The script calculates the PGA value to be equal to the selected
''' record's Best_Fit1 value.  It also calculates the AcceptFlag with
''' the current date and rebuilds (refreshes) the listbox from 'newfile'
''' to reflect changes.
'''
''' Scripts Called:
'''
'''         MyDialog.Lbt_Apply.Update by broadcast
'''         MyDialog.Lbt_Accept.Update by broadcast
'''         MyDialog.Lbt_UnAccept.Update by broadcast
'''         MyDialog.GroupButtons.Update by broadcast
'''
''' Notes:  The CovSysID (stations' unique id attribute) is located
'''       inside the listbox by location.  It is at location '0', or the
'''       first attribute inside the listbox.  If the location of
'''       CovSysId is moved, then THE SELECTION NUMBER MUST BE CHANGED!
'''
'''       The _aListBox global variable was set to the Lbx_Newfile
'''       listbox in the script MyDialog.Open.
'''
''' ------------------------------------------------------------------------

''' Issue a broadcast to all of the buttons inside the dialog
''' box to run their update scripts (listed above).  This will enable the
''' buttons.
self.BroadcastUpdate

''' Now select records in newfile by CovSysID field value, the first
''' field inside the listbox (the first field is always '0', SEE NOTES
''' ABOVE)

''' First get the covsysid value from the listbox
aList = _aListBox.GetSelection
aVal = (aList.Get(0))

''' Now query newfile to find the associated record
aVTab = av.GetProject.FindDoc("newfile").GetVTab
aBitmap = aVTab.GetSelection
aFieldList = aVTab.GetFields
aField = aFieldList.Get(0).AsString
q1 = "( ["+aField+"] = "+aVal+" )"
aVTab.Query( q1, aBitmap, #VTAB_SELTYPE_NEW)
aVTab.UpdateSelection

''' Now it's time to calculate the new PGA value
aVTab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("PGA")

''' Populate a number field with the value of the selected record's
''' BestFit_1 value.
theBF1List = _aListBox.GetSelection
aVal = (theBF1List.Get(4))
aVTab.Calculate( aVal, aField)
aVTab.Refresh
aVTab.SetEditable(False)

''' Populate today's date (as a number) in for AcceptFlag
aField = aVTab.FindField("AcceptFlag")
today = Date.Now
```

```
today.SetFormat( "yyyyMMdd" )
aStr = today.AsString
aVTab.Calculate(aStr, aField)

'"Recreate listbox (refresh to show changes)
aVTab = av.GetProject.FindDoc("newfile").GetVTab
myFields = aVTab.GetFields
self.DefineFromVTab(aVTab, myFields, false)
self.FitColumns(0..10, true)

'" The dialog box will scroll to the top of the list,
'" even though another cell may be selected.  To avoid
'" this from happening, I set the current cell to the
'" value of the CovSysID field from the VTab and set the
'" listbox cell with that value as the current cell.  Once
'" the current cell is selected, it can be scrolled into view
'" with the FindByValue request
'" Unfortunately, the same could not be done to hold the horizontal
'" scroll for the listbox selection type (enum LISTBOX_SELECTION_SINGLEROW)
aVTab = av.GetProject.FindDoc("newfile").GetVTab
aField = aVTab.FindField("CovSysID")
for each rec in aVTab.GetSelection
  aVal = aVTab.ReturnValueString (aField, rec)
  aCurrentFld = _aListBox.GetCurrentField.AsString
  _aListBox.FindByValue (aVal)
  _aListBox.ShowCurrent
end
```

```
'''
''' MyDialog.Lbx_Newfile.Select
'''
''' smm 04-99
'''
''' Runs when record in listbox is selected (or clicked).  It is
''' associated with the listbox's SELECT property.
'''
''' Scripts Called:
'''
'''         MyDialog.Lbt_Accept.Update by broadcast
'''         MyDialog.Lbt_UnAccept.Update by broadcast
'''         MyDialog.Lbt_Apply.Update by broadcast
'''         MyDialg.GroupButtons.Update by broadcast
'''
''' The script first selects the corresponding record in 'newfile' so
''' that only the correct record is written to whenever an editing action
''' takes place inside the PGA Update dialog box (e.g. clicking the
''' ACCEPT or UNACCEPT buttons, any of the group buttons, or double-clicking
''' a listbox's record).
'''
''' ------------------------------------------------------------------------

''' When a record in the listbox is selected, the associated record in
''' newfile needs to select as well.  Select records in newfile by CovSysID
''' field value.
aList = _aListBox.GetSelection
aVal = (aList.Get(0))
aVTab = av.GetProject.FindDoc("newfile").GetVTab
aBitmap = aVTab.GetSelection
aFieldList = aVTab.GetFields
aField = aFieldList.Get(0).AsString
q1 = "( ["+aField+"] = "+aVal+" )"
aVTab.Query( q1, aBitmap, #VTAB_SELTYPE_NEW)
aVTab.UpdateSelection

''' A broadcast is sent to enable the listbox's listener components.
self.BroadcastUpdate
```

```
'''
''' MyDialog.Lbx_Newfile.Update
'''
''' smm 04-99
'''
''' Runs whenever a broadcast is sent from any of the following scripts:
'''
'''         MyDialog.Lbt_Accept.Update
'''         MyDialog.Lbt_UnAccept.Update
'''         NewfileUpdate1
'''         NewfileUpdate2
'''         NewfileUpdate3
'''         NewfileUpdate4
'''         NewfileUpdate5
'''         NewfileUpdate6
'''         NewfileUpdate7
'''
''' This script is associated to the Lbx_Newfile listbox's UPDATE property.
'''
''' The script executes a refresh of the listbox whenever a write is
''' issued to newfile by rebuilding the listbox from newfile to reflect
''' any changes that may have taken place. The refresh occurs whenever
''' the ACCEPT, UNACCEPT, or any of the group buttons are clicked.
'''
''' Notes:  The global variable _aListBox is set to Lbx_Newfile during the
'''         execution of MyDialog.Open
'''
''' ----------------------------------------------------------------------------

'''Recreate the listbox from newfile
aVTab = av.GetProject.FindDoc("newfile").GetVTab
myFields = aVTab.GetFields
self.DefineFromVTab(aVTab, myFields, false)
self.FitColumns(0..10, true)

''' The dialog box will scroll to the top of the list,
''' even though another cell may be selected.  To avoid
''' this from happening, I set the current cell to the
''' value of the CovSysID field from the VTab and set the
''' listbox cell with that value as the current cell.  Once
''' the current cell is selected, it can be scrolled into view
''' with the FindByValue request.
''' Unfortunately, the same could not be done to hold the horizontal
''' scroll for the listbox selection type (enum LISTBOX_SELECTION_SINGLEROW)
aVTab = av.GetProject.FindDoc("newfile").GetVTab
aField = aVTab.FindField("CovSysID")
for each rec in aVTab.GetSelection
  aVal = aVTab.ReturnValueString (aField, rec)
  aCurrentFld = _aListBox.GetCurrentField.AsString
  _aListBox.FindByValue (aVal)
  _aListBox.ShowCurrent
end
```

```
'''
''' MyDialog.Open
'''
''' smm 04-99
'''
''' Executes when called from the following scripts:
'''
'''         MyDialog.ExportVTab.FromTable
'''         MyDialog.ExportVTab.FromView
'''
''' Scripts Called:
'''
'''         MyProject.LabelCovSysID
'''
''' This script populates the listbox Lbx_Newfile inside the PGA Update
''' Dialog Box with the values from newfile, sets the broadcast/listener
''' controls within the dialog box, sets the enabling feature
''' of each button (except CANCEL) to false by default.
'''
''' ---------------------------------------------------------------------

''' Run MyProject.LabelCovSysID to label the PTC stations that are within
''' the SGAA View extent.
av.Run("MyProject.LabelCovSysID",nil)

''' First, we'll set some control variables.  The dialog box consists of ten buttons
''' and a listbox.  Each control's name describes the object.  The label buttons
''' start with lbt_ and the listbox with lbx_.  The prefix is followed by how the
''' button is labeled.  For example, lbt_Accept is the name of the Accept button,
''' lbt_Grp1 is the Group 1 (labeled '1') button, etc.
theDialog = av.GetProject.FindDialog("PGA_Update")
_aListBox = self.FindByName("lbx_Newfile")
Lbt_Accept = self.FindByName("lbt_Accept")
Lbt_Apply = self.FindByName("lbt_Apply")
aListBox = self.FindByName("lbx_Newfile")
Lbt_Grp1 = self.FindByName("lbt_Grp1")
Lbt_Grp2 = self.FindByName("lbt_Grp2")
Lbt_Grp3 = self.FindByName("lbt_Grp3")
Lbt_Grp4 = self.FindByName("lbt_Grp4")
Lbt_Grp5 = self.FindByName("lbt_Grp5")
Lbt_Grp6 = self.FindByName("lbt_Grp6")
Lbt_Grp7 = self.FindByName("lbt_Grp7")
Lbt_UnAccept = self.FindByName("lbt_UnAccept")

''' Populate the listbox with the values from newfile, the table derived from
''' the selected set in Attributes of PTC Stations
aVTab = av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(true)
for each rec in aVTab
  aField = (aVTab.FindField("PGA"))

'''Some records in Table1 may not get matched to those in the attribute table.  This
'''causes an error if the station is inadvertently selected and an edit is
'''attempted from the dialog box (since the station is not joined to any record in
'''Table1, the attribute virtually does not exist).  So, we're going to check to see
'''if the record returns a "Number Nill" string when the PGA (aField) attribute
'''is called:

  if (aVTab.ReturnValue(aField,rec).AsString = "Number Null") then

''' and if the record returns the string "Number Nill", then we'll remove it from our
''' listbox:

    aVTab.RemoveRecord(rec)
  end
end

'''Now turn off the editing mode of newfile
```

```
aVTab.SetEditable(false)

'" It's time to get all of the fields from newfile and use their values to
'" populate the listbox

myFields = aVTab.GetFields
aListBox.DefineFromVTab(aVTab, myFields, false)
aListBox.FitColumns(0..10, true)

'" Set the controls as listeners and/or broadcasters.  The
'" request SetListners sets the preceding button as the broadcaster and all
'" specified buttons as its listeners.  For example, the Lbt_Grp1 button broadcasts to
'" aListBox (the listbox) by the request Lbt_Grp1.SetListeners({aListBox}).
'" Whenever a broadcast is sent from Lbt_Grp1, the listbox runs its UPDATE script.

aListBox.SetListeners({Lbt_Accept, Lbt_UnAccept, Lbt_Apply, Lbt_Grp1, Lbt_Grp2, Lbt_Grp3, Lbt_Grp4, Lbt_Grp5, Lbt_C
Lbt_Grp1.SetListeners({aListBox})
Lbt_Grp2.SetListeners({aListBox})
Lbt_Grp3.SetListeners({aListBox})
Lbt_Grp4.SetListeners({aListBox})
Lbt_Grp5.SetListeners({aListBox})
Lbt_Grp6.SetListeners({aListBox})
Lbt_Grp7.SetListeners({aListBox})
Lbt_Accept.SetListeners({aListBox, Lbt_UnAccept})
Lbt_UnAccept.SetListeners({aListBox, Lbt_Accept})

'" Set enabled to false for the APPLY, ACCEPT, UNACCEPT, and the group buttons.
'" These buttons will initially be greyed out until the listbox sends a broadcast
'" to each button to run its UPDATE script.

Lbt_Accept.SetEnabled(false)
Lbt_Apply.SetEnabled(false)
Lbt_Grp1.SetEnabled(false)
Lbt_Grp2.SetEnabled(false)
Lbt_Grp3.SetEnabled(false)
Lbt_Grp4.SetEnabled(false)
Lbt_Grp5.SetEnabled(false)
Lbt_Grp6.SetEnabled(false)
Lbt_Grp7.SetEnabled(false)
Lbt_UnAccept.SetEnabled(false)
```

```
'''-------------------------------------------------------
''' MyLegend.Load
'''
''' smm & jlh 05-99
'''
''' Script is run from MyProject.Startup
'''
''' This script loads the legend file seasonal.avl
''' into the view to classify PTC Stations
''' by their Classify field values. It also
''' loads rd24k.avl to classify 1997 road coverage
'''
''' UPDATES TO SOME OF THESE FILENAMES IN CODE WILL
''' BE NECESSARY WHEN THE rd & ptc FILES ARE UPDATED
'''-------------------------------------------------------

seaslegfile  = "d:\seasonal\ptc.avl".AsFileName
nhs97legfile = "o:\Arcdata\Legends\Division\nhs97all.avl".AsFileName
rd97legfile  = "o:\Arcdata\Legends\Division\rd24k.avl".AsFileName
'''rd98legfile = "o:\Arcdata\Legends\Division\?????".AsFileName

aView = av.GetProject.FindDoc("NCSU-ITRE Seasonal Group Assignment Application")
theThemes = aView.GetThemes

for each t in theThemes
  """msgbox.info("The current Theme = " + t.Getname, "")
  if (t.GetName = "PTC Stations") then
      ''' PTC Stations theme will need to change source files (eventually)
      ''' right now it runs off of ts-div5.shp, need to replace it with
      ''' a statewide file when digitizing is completed.
     aSeasPTCLegend = Legend.Make(#SYMBOL_MARKER)          ''' new empty legend
     aSeasPTCLegend.Load(seaslegfile, #LEGEND_LOADTYPE_ALL)  ''' fill it w/ saved info
     t.SetLegend (aSeasPTCLegend)                            ''' apply it to the theme
     t.InvalidateLegend                                ''' refresh the theme
  end
  if (t.GetSrcName.GetName = "rd.shp") then
       ''' will need to change the legend file below
       ''' when new rds files come online
     aRd24kLegend = Legend.Make(#SYMBOL_PEN)               ''' new empty legend
     aRd24kLegend.Load(rd97legfile, #LEGEND_LOADTYPE_ALL)    ''' fill it w/ saved info
     t.SetLegend (aRd24kLegend)                              ''' apply it to the theme
     t.InvalidateLegend                                ''' refresh the theme
  end
  if (t.GetName = "National Highway System") then
      ''' NHS theme will need to change source files
      ''' right now it runs off of nhs97all.shp, need to
      ''' replace it with a current file every year.
     anNHSLegend = Legend.Make(#SYMBOL_PEN)                ''' new empty legend
     anNHSLegend.Load(nhs97legfile, #LEGEND_LOADTYPE_ALL)    ''' fill it w/ saved info
     t.SetLegend (anNHSLegend)                               ''' apply it to the theme
     t.InvalidateLegend                                ''' refresh the theme
  end

end



'MsgBox.Report("DataSource = " + aSrcName.getDataSource.AsString +nl+
'            "FileName = "  + aSrcName.getFileName.AsString +nl+
'            "Name = "      + aSrcNAme.getName.Asstring +nl+
'            "SubName = "   + aSrcName.getsubname.asstring, "")
```

```
'''
''' MyProject.LabelCovSysID
'''
''' smm 06-21-99
'''
''' When evaluating multiple selections in the dialogue
''' box, the station numbers are displayed on screen to
''' delineate one from another.  This script creates those
''' labels whenever the dialog box is opened.
'''
''' Script run from:  MyDialog.Open
'''
'''-----------------------------------------------------------

aView = av.GetActiveDoc
aTheme = aView.GetActiveThemes.Get(0)
aExt = aView.GetDisplay.ReturnVisExtent
aLabeler = Labeler.Make(aExt)

''' These weights are the defaults for a line theme.

aLabeler.SetFeatureWeight(#LABEL_WEIGHT_NO)
aLabeler.SetLabelWeight(#LABEL_WEIGHT_HIGH)

aLabeler.RemoveDuplicates(true)

''' Set the textsymbol for the theme you are labeling

aTextSym = TextSymbol.Make
aTextSym.SetFont(Font.Make("Times","Bold"))
aTextSym.SetSize(8)
aTheme.SetLabelTextSym(aTextSym)

''' This starts the labeling process and corresponds to the first
''' status bar you see on the application window.

aLabeler.Load(aTheme)

''' Get the Labels from the labeler (corresponds to the second
''' status bar you see on the application window) and draw them
''' on the view.
''' The GetAutoLabels request automatically adds the new labels
''' to the aView's GraphicList and aTheme's GraphicSet.

aView.GetAutoLabels(aLabeler, false)
```

```
'''
''' MyProject.RemoveLabels
'''
''' smm 06-21-99
'''
''' Removes the CovSysID labels created during the opening
''' of the dialog box.  Run from MyDialog.Close
''' ------------------------------------------------------

v = av.GetActiveDoc
for each t in v.GetVisibleThemes
  if (t.IsActive) then
    if (t.GetGraphics.HasLabels) then
       t.GetGraphics.SelectLabels
     t.GetGraphics.Invalidate
 end
    end
end
v.GetGraphics.ClearSelected
```

```
'''
''' MyProject.ShutDown
'''
''' smm 04-99
'''
''' This script is called whenever the project is closed.  It is
''' assigned as the project property's ShutDown script.
'''
''' The script is two-part:
'''     PART 1:  Captures the SGAA View extent and returns the
'''              variables to the ASCII file "sapextents.txt"
'''              (refer to MyView.MapExtent for the retrieval of
'''              these values).
'''     PART 2:  Checks for "qtable" and deletes it if it exists.
'''              The table "qtable" is created during the project
'''              startup (refer to MyProject.StartUp and
'''              Query.Table1).
'''
''' Notes:  The pathname for "sapextents.txt" may need to be changed
'''         when placing this project on a different drive.
'''
''' ----------------------------------------------------------------

''' First capture the SGAA View and return its extent variables.
''' Refer to MyView.MapExtent for the retrieval of these values.

theView = av.FindDoc("NCSU-ITRE Seasonal Group Assignment Application")
aDisplay = theView.GetDisplay
theRect = aDisplay.ReturnExtent
theOrigin = theRect.ReturnOrigin
theSize = theRect.ReturnSize

theOriginX = theOrigin.GetX
theOriginY = theOrigin.GetY
theSizeX = theSize.GetX
theSizeY = theSize.GetY

''' Create/recreate/rewrite the file "sapextents.txt" to contain
''' the extent variables above for retrieval during startup.
''' WARNING!!!  THE PATHNAME FOR SAPEXTENTS.TXT MAY NEED TO BE
'''             CHANGED WHEN PLACING THIS PROJECT ON ANOTHER DRIVE!!!

aTextFile = LineFile.Make("c:\seasonal\pfiles\sapextents.txt".AsFileName, #FILE_PERM_WRITE)
aTextFile.WriteElt(theOriginX.AsString)
aTextFile.WriteElt(theOriginY.AsString)
aTextFile.WriteElt(theSizeX.AsString)
aTextFile.WriteElt(theSizeY.AsString)
aTextFile.Close

''' Check to see if qtable exists.  If it does, delete it before
''' closing.

aVTab = av.GetProject.FindDoc("qtable")
if (aVTab.AsString = "qtable") then
  av.GetProject.RemoveDoc(aVTab)
end
```

```
'''
''' MyProject.StartUp
'''
''' smm 04-99
'''
''' Executes at project start-up (is associated with the
''' project's Project StartUp property).
'''
''' Scripts Called:
'''
'''         MyView.MapExtent
'''         Date.Warning
'''         Query.Table1
'''         MyLegend.Load
'''
''' Sets up initial project parameters by running associated scripts
''' (listed above) and setting global variables that are used
''' throughout the project.  It also checks to ensure that Table1
''' is present and will send the user an error message if it does not.
'''
''' Notes:  The global variable PASSWORD is set at the beginning
'''         of this script.  It only needs to be changed here, and
'''         in no other script.  If changed, please MAKE SURE THE
'''         NEW PASSWORD IS SURROUNDED BY DOUBLE QUOTES!
'''
''' ---------------------------------------------------------------------

''' Set up the global variable to get the environment variable SEASCOV
_seasonal = "SEASCOV"

''' Set the global variable password.
''' WARNING:  Be sure to put the string in DOUBLE
'''           QUOTES or the password will fail.

_thePass = "admin"

''' Check to see if Table1 is still around.  If not
''' send an error message and prompt for a password.  If the
''' password fails, the project closes.

errorVTab = av.GetProject.FindDoc("Table1").AsString
if ((errorVTab = "Table1").Not) then
  MsgBox.Error ("Table1 does not exist!  Please consult your systems administrator.  This project will now shut down.","")
  aPass = MsgBox.Password
  if (aPass = _thePass) then
    MsgBox.Info("You're In!!!","")

  else
    MsgBox.Error("Your password is incorrect.  This project will close.  Please consult your systems administrator.","")
    theProject = av.GetProject
    theProject.Close
    theProject = nil
  end
end

''' Run Query.Table1 which asks the user to select the county or
''' counties with which he or she wishes to work

av.Run("Query.Table1", nil)

''' Run MyLegend.Load to load the legend files (refer to MyLegend.Load)

av.Run("MyLegend.Load", nil)

''' Return the user to the view extent at the last shutdown.

av.Run("MyView.MapExtent", nil)

''' Before we start, we need to check on the date.  If the reset
```

''' date has passed, the user will be prompted for a password.
''' If the password fails, or none is given, the project closes with
''' a message.

av.Run("Date.Warning", nil)

```
'''
''' MyView.Accept
'''
''' smm 05-99
'''
''' Runs when the ACCEPT PGA button located on the View window interface is
''' clicked (associated with the ACCEPT PGA button's CLICK property).
'''
''' This script allows the user a shortcut to accepting stations without
''' having to open the dialog box.  The edits are made directly to both the
''' Table1 and QTABLE AcceptFlag attributes.  The resulting value is the
''' current date.
'''
''' ------------------------------------------------------------------------

''' Check to see if the correct theme is active.
''' If not, prompt the user with an Error Message.
theTable = av.GetProject.FindDoc("Attributes of PTC Stations")
aView = av.GetProject.FindDoc("NCSU-ITRE Seasonal Group Assignment Application")
theThemeCheck = aView.GetActiveThemes
aStr = "PTC Stations"
for each i in theThemeCheck
  if ((i.AsString = aStr).Not) then
    MsgBox.Error("You must make PTC Stations an active theme!","")

''' If PTC Stations is active, we continue on...

''' First, create newfile and set its name to newfile
  else
    theVTab = theTable.GetVTab
    ' create a new dBASE file of the table's selected records
    theVTab.Export ( "newfile".asFileName, dBASE, TRUE )
    ' output file, newfile.dbf, contains only the selected records

    theFileName = "newfile.dbf".asFileName
    theVTab = VTab.Make (theFileName, false, false)
    theTable = Table.Make(theVTab)
    'theTable.GetWin.Open
    theName = theTable.SetName ("newfile")

''' Now make the table editable and grab the field AcceptFlag
    aVtab =  av.GetProject.FindDoc("newfile").GetVTab
    aVTab.SetEditable(True)
    aField = aVTab.FindField("AcceptFlag")

''' We're going to calculate the current date into the selected stations'
''' AcceptFlag, so we have to capture the date and return it as a string.
    today = Date.Now
    today.SetFormat( "yyyyMMdd" )
    aStr = today.AsString

''' Now that we have the date, we can calculate it in for AcceptFlag
    aVTab.Calculate(aStr, aField)

''' Now make newfile un-editable
    aVTab.SetEditable(False)

''' Set the table variables and execute the SQL connection
    aVTab =  av.GetProject.FindDoc("newfile").GetVTab
    aTable = av.GetProject.FindDoc("Table1")
    theSQLCon = SQLCon.Find(_seasonal)
    theSQLCon.Login("")

''' For each record selected (each station selected) we'll calculate the new AcceptFlag
''' value from newfile and a new value for Classify to reflect the changes in the view.
    for each rec in aVTab
      aRelateField = aVTab.FindField("CovSysID")
      aJoinVal = aVTab.ReturnValueString(aRelateField,rec)
      anAcceptField = aVTab.FindField("AcceptFlag")
```

```
    anAcceptVal = aVTab.ReturnValueString(anAcceptField,rec)
    anExpression = "(((tblSeasonalFactorStations.CovSysID) =" ++aJoinVal++ "))"
    anAccessFlg = aVTab.FindField("AccessFlag")
    anAccessVal = aVTab.ReturnValue(anAccessFlg,rec)
    aPGAField = aVTab.FindField("PGA")
    aPGAVal = aVTab.ReturnValueString(aPGAField,rec)
    aPGAValNum = aVTab.ReturnValue(aPGAField,rec)

''' Update Table1 with the new AcceptFlag values from newfile
    UpdateAcceptFlg = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.AcceptFlag =" ++anAcceptV
    theSQLCon.ExecuteSQL(UpdateAcceptFlg)

''' Calculate the Classify Field:

''' The original values of Classify is the same as that of the PGA.  However, the number
''' is multiplied by either -1, 100, or both if certain situations occur:
'''
'''    If the AccessFlag is '0' then the value will be multiplied by -1,
'''    If the AccessFlag is greater than '0', the number will remain positive,
'''    If the AcceptFlag is greater than '0', i.e. the station has been 'accepted', then
'''    the Classify value (negative or positive) is multiplied by 100.
'''
''' This gives four possible scenerios of which we'll classify the PTC Stations theme by:
'''  1. Station has data but is not accepted (1-7)
'''  2. Station has data and is accepted (100-700)
'''  3. Station has no data and is not accepted ([-1]-[-7])
'''  4. Station has no data and is accepted ([-100]-[-700])

    if (anAccessVal = "0") then
     if (anAcceptVal = "0") then
       aNegPGAVal = (aPGAValNum * -1).AsString
       UpdateClass = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.Classify =" ++aNegPGAVal++
       theSQLCon.ExecuteSQL(UpdateClass)
     else
       aNegPGAVal = (aPGAValNum * -100).AsString
       UpdateClass = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.Classify =" ++aNegPGAVal++
       theSQLCon.ExecuteSQL(UpdateClass)
     end
    else
     if (anAcceptVal = "0") then
       UpdateClass = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.Classify =" ++aPGAVal++ "Wh
       theSQLCon.ExecuteSQL(UpdateClass)
     else
       aPGAVal = (aPGAValNum * 100).AsString
       UpdateClass = "UPDATE tblSeasonalFactorStations SET tblSeasonalFactorStations.Classify =" ++aPGAVal++ "Wh
       theSQLCon.ExecuteSQL(UpdateClass)
     end
    end

  end

''' Now write the edits to qtable so the changes will be reflected if the
''' dialog box is opened.

  aVTab =  av.GetProject.FindDoc("newfile").GetVTab
  aTable = av.GetProject.FindDoc("qtable").GetVTab

  for each rec in aVTab
    aRelateField = aVTab.FindField("CovSysID")
    aJoinVal = aVTab.ReturnValueString(aRelateField,rec)
    aPGAField = aVTab.FindField("PGA")
    aPGAVal = aVTab.ReturnValueString(aPGAField,rec)
    aPGAValNum = aVTab.ReturnValue(aPGAField,rec)
    anAcceptField = aVTab.FindField("AcceptFlag")
    anAcceptVal = aVTab.ReturnValueString(anAcceptField,rec)
    anAccessFlg = aVTab.FindField("AccessFlag")
    anAccessVal = aVTab.ReturnValue(anAccessFlg,rec)

    aTable.SetEditable(True)
```

al++ "WHERE" ++anExpression++ ";"

"WHERE" ++anExpression++ ";"

"WHERE" ++anExpression++ ";"

HERE" ++anExpression++ ";"

HERE" ++anExpression++ ";"

```
    aBitmap = aTable.GetSelection
    theExpression = "([CovSysID] = "+aJoinVal+")"
    aTable.Query( theExpression, aBitmap, #VTAB_SELTYPE_NEW)
    aTable.UpdateSelection

    aField = aTable.FindField("AcceptFlag")
    aTable.Calculate(anAcceptVal, aField)

    aField = aTable.FindField("Classify")
    if (anAccessVal = "0") then
      if (anAcceptVal = "0") then
        aNegPGAVal = (aPGAValNum * -1).AsString
        aTable.Calculate(aNegPGAVal, aField)
      else
        aNegPGAVal = (aPGAValNum * -100).AsString
        aTable.Calculate(aNegPGAVal, aField)
      end
    else
      if (anAcceptVal = "0") then
        aTable.Calculate(aPGAVal, aField)
      else
        aPGAVal = (aPGAValNum * 100).AsString
        aTable.Calculate(aPGAVal, aField)
      end
    end
    aTable.SetEditable(False)

'" Update Attributes of PTC Stations table to display changes made
'" to Table1

    theTable = av.getproject.finddoc("Attributes of PTC Stations")
    theVTab = theTable.GetVTab
    theVTab.Refresh

'" Now delete 'newfile'

    theProject = av.GetProject
    aTableDelete = theProject.FindDoc("newfile")
    av.GetProject.RemoveDoc( aTableDelete )
  end
end
```

```
'''
''' MyView.HotLink
'''
''' smm 05-99
'''
''' Runs when the Station SAS Analysis Image tool is selected and
''' the user clicks on a PTC station inside the SGAA View.  It is
''' associated with the button's APPLY field.
'''
''' This customized hotlink associates the station that the user
''' clicks on with its SAS output image file.  The image pathname
''' is built on-the-fly to eliminate the need for storing the name
''' inside the attribute table.  If the files change directories,
''' only this script needs to change to reflect the new pathname.
'''
''' Notes:  The PTC Stations theme MUST BE ACTIVE in order for
'''         tool to work.
'''
'''         If the pathname of the images changes, THE PATHNAME
'''         IN THIS SCRIPT FOR THE FILENAME MUST CHANGE AS WELL!

theView = av.GetActiveDoc
found = false
p = theView.GetDisplay.ReturnUserPoint
for each t in theView.GetActiveThemes
   recs = t.FindByPoint(p)
   for each rec in recs
    theFTab = t.GetFTab
    theField = theFTab.FindField("CovSysID")
    found = true
    theVal = t.ReturnValueString(theField.GetName, rec)
    if (not (theVal.IsNull)) then
'''  WARNING:  If the files are placed in a new directory, the
'''            following pathname needs to be changed!!!
      theVal = "c:\seasonal\images\htimages\station"+theVal+".gif"
      if (File.Exists(theVal.AsFileName)) then
        i = ImageWin.Make(theVal.AsFileName, theVal)
        i.SetScaled(false)
        i.Open
        i.Resize(650,700)
        i.MoveTo(375,50)
      else
        MsgBox.Warning("File "+theVal+" not found.","Hot Link")
      end
    end
   end
end
if (not found) then
  System.Beep
end
```

```
'''
''' MyView.MapExtent
'''
''' smm 4-99
'''
''' revised jlh 4-99, smm 4-99
'''
''' This script is called from MyProject.StartUp and restores
''' the map extents used in the previous working session, which are
''' saved in an ASCII text file in the /seasonal directory.
''' Filename = 'sapextents.txt'
''' format:  x origin
'''          y origin
'''          x size
'''          y size
'''
''' Notes:  The file pathname for "sapextents.txt" may need to be changed
'''         if loading the project on a different drive.
''' ----------------------------------------------------------------------

theDoc = av.FindDoc("NCSU-ITRE Seasonal Group Assignment Application")
theDisplay = theDoc.GetDisplay

''' WARNING!!!  PATHNAME MAY CHANGE IF LOADING THIS PROJECT ONTO A
'''             DIFFERENT DRIVE!!!
tf = LineFile.Make("c:\seasonal\pfiles\sapextents.txt".AsFileName, #FILE_PERM_READ)

theOriginX = tf.ReadElt.AsNumber
theOriginY = tf.ReadElt.AsNumber
theSizeX = tf.ReadElt.AsNumber
theSizeY = tf.ReadElt.AsNumber

''' Error checking to see if the extent values are valid.  If not, zoom
''' to the extent of the entire state.  A valid extent would include
''' anything within the stateplane NAD83 (meters) statewide extent
''' of the county boundary shapefile cb.shp.

if (theOriginX.>(0) and theOriginY.>(0)) then
  if (theSizeX.<(887000) and theSizeY.<(340000)) then
    aRect = Rect.Make(theOriginX@theOriginY, theSizeX@theSizeY)
    theDoc.GetWin.Open
    theDisplay.ZoomToRect(aRect)
  end
else
  r = theDoc.ReturnExtent
  if (r.IsEmpty) then
    return nil
  elseif (r.ReturnSize = (0@0) ) then
    theDoc.GetDisplay.PanTo(r.ReturnOrigin)
  else
    theDoc.GetDisplay.SetExtent(r.Scale(1.1))
    av.GetProject.SetModified(true)
  end
end
```

```
'''
''' NewfileUpdate1
'''
''' smm 04-99
'''
''' Executes when the PGA Group button '1' (Lbt_Grp1) is
''' clicked (associated with that button's CLICK property).
'''
''' Scripts Called:
'''
'''         MyDialog.Lbx_Newfile.Update by broadcast
'''
''' Calculates a value of '1' into newfile for the selected
''' record in the dialog box's listbox.  It also calculates
''' the AcceptFlag field with the current date.
'''
''' ----------------------------------------------------------

aVtab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("PGA")

'''Populate the PGA field with the value 1
aVTab.Calculate("1", aField)
aVTab.Refresh
aVTab.SetEditable(False)

'''Populate today's date (as a number) in for AcceptFlag
aField = aVTab.FindField("AcceptFlag")
today = Date.Now
today.SetFormat( "yyyyMMdd" )
aStr = today.AsString
aVTab.Calculate(aStr, aField)

'''Broadcast to the listbox Lbx_Newfile to rebuild from
''' newfile (refreshes the listbox)

 self.BroadcastUpdate
```

```
'''
''' NewfileUpdate2
'''
''' smm 04-99
'''
''' Executes when the PGA Group button '2' (Lbt_Grp2) is
''' clicked (associated with that button's CLICK property).
'''
''' Scripts Called:
'''
'''         MyDialog.Lbx_Newfile.Update by broadcast
'''
''' Calculates a value of '2' into newfile for the selected
''' record in the dialog box's listbox.  It also calculates
''' the AcceptFlag field with the current date.
'''
''' ----------------------------------------------------------

aVtab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("PGA")

'''Populate the PGA field with the value 2
aVTab.Calculate("2", aField)
aVTab.Refresh
aVTab.SetEditable(False)

'''Populate today's date (as a number) in for AcceptFlag
aField = aVTab.FindField("AcceptFlag")
today = Date.Now
today.SetFormat( "yyyyMMdd" )
aStr = today.AsString
aVTab.Calculate(aStr, aField)

'''Broadcast to the listbox Lbx_Newfile to rebuild from
''' newfile (refreshes the listbox)

 self.BroadcastUpdate
```

```
'''
''' NewfileUpdate3
'''
''' smm 04-99
'''
''' Executes when the PGA Group button '3' (Lbt_Grp3) is
''' clicked (associated with that button's CLICK property).
'''
''' Scripts Called:
'''
'''         MyDialog.Lbx_Newfile.Update by broadcast
'''
''' Calculates a value of '3' into newfile for the selected
''' record in the dialog box's listbox.  It also calculates
''' the AcceptFlag field with the current date.
'''
''' ----------------------------------------------------------

aVtab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("PGA")

'''Populate the PGA field with the value 3
aVTab.Calculate("3", aField)
aVTab.Refresh
aVTab.SetEditable(False)

'''Populate today's date (as a number) in for AcceptFlag
aField = aVTab.FindField("AcceptFlag")
today = Date.Now
today.SetFormat( "yyyyMMdd" )
aStr = today.AsString
aVTab.Calculate(aStr, aField)

'''Broadcast to the listbox Lbx_Newfile to rebuild from
''' newfile (refreshes the listbox)

 self.BroadcastUpdate
```

```
'''
''' NewfileUpdate4
'''
''' smm 04-99
'''
''' Executes when the PGA Group button '4' (Lbt_Grp4) is
''' clicked (associated with that button's CLICK property).
'''
''' Scripts Called:
'''
'''         MyDialog.Lbx_Newfile.Update by broadcast
'''
''' Calculates a value of '4' into newfile for the selected
''' record in the dialog box's listbox.  It also calculates
''' the AcceptFlag field with the current date.
'''
''' ----------------------------------------------------------

aVtab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("PGA")

'''Populate the PGA field with the value 4
aVTab.Calculate("4", aField)
aVTab.Refresh
aVTab.SetEditable(False)

'''Populate today's date (as a number) in for AcceptFlag
aField = aVTab.FindField("AcceptFlag")
today = Date.Now
today.SetFormat( "yyyyMMdd" )
aStr = today.AsString
aVTab.Calculate(aStr, aField)

'''Broadcast to the listbox Lbx_Newfile to rebuild from
''' newfile (refreshes the listbox)

self.BroadcastUpdate
```

```
'''
''' NewfileUpdate5
'''
''' smm 04-99
'''
''' Executes when the PGA Group button '5' (Lbt_Grp5) is
''' clicked (associated with that button's CLICK property).
'''
''' Scripts Called:
'''
'''         MyDialog.Lbx_Newfile.Update by broadcast
'''
''' Calculates a value of '5' into newfile for the selected
''' record in the dialog box's listbox.  It also calculates
''' the AcceptFlag field with the current date.
'''
''' ----------------------------------------------------------

aVtab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("PGA")

'''Populate the PGA field with the value 5
aVTab.Calculate("5", aField)
aVTab.Refresh
aVTab.SetEditable(False)

'''Populate today's date (as a number) in for AcceptFlag
aField = aVTab.FindField("AcceptFlag")
today = Date.Now
today.SetFormat( "yyyyMMdd" )
aStr = today.AsString
aVTab.Calculate(aStr, aField)

'''Broadcast to the listbox Lbx_Newfile to rebuild from
''' newfile (refreshes the listbox)

self.BroadcastUpdate
```

```
'''
''' NewfileUpdate6
'''
''' smm 04-99
'''
''' Executes when the PGA Group button '6' (Lbt_Grp6) is
''' clicked (associated with that button's CLICK property).
'''
''' Scripts Called:
'''
'''         MyDialog.Lbx_Newfile.Update by broadcast
'''
''' Calculates a value of '6' into newfile for the selected
''' record in the dialog box's listbox.  It also calculates
''' the AcceptFlag field with the current date.
'''
''' ----------------------------------------------------------

aVtab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("PGA")

'''Populate the PGA field with the value 6
aVTab.Calculate("6", aField)
aVTab.Refresh
aVTab.SetEditable(False)

'''Populate today's date (as a number) in for AcceptFlag
aField = aVTab.FindField("AcceptFlag")
today = Date.Now
today.SetFormat( "yyyyMMdd" )
aStr = today.AsString
aVTab.Calculate(aStr, aField)

'''Broadcast to the listbox Lbx_Newfile to rebuild from
''' newfile (refreshes the listbox)

 self.BroadcastUpdate
```

```
'''
''' NewfileUpdate7
'''
''' smm 04-99
'''
''' Executes when the PGA Group button '7' (Lbt_Grp7) is
''' clicked (associated with that button's CLICK property).
'''
''' Scripts Called:
'''
'''         MyDialog.Lbx_Newfile.Update by broadcast
'''
''' Calculates a value of '7' into newfile for the selected
''' record in the dialog box's listbox.  It also calculates
''' the AcceptFlag field with the current date.
'''
''' ----------------------------------------------------------

aVtab =  av.GetProject.FindDoc("newfile").GetVTab
aVTab.SetEditable(True)
aField = aVTab.FindField("PGA")

'''Populate the PGA field with the value 7
aVTab.Calculate("7", aField)
aVTab.Refresh
aVTab.SetEditable(False)

'''Populate today's date (as a number) in for AcceptFlag
aField = aVTab.FindField("AcceptFlag")
today = Date.Now
today.SetFormat( "yyyyMMdd" )
aStr = today.AsString
aVTab.Calculate(aStr, aField)

'''Broadcast to the listbox Lbx_Newfile to rebuild from
''' newfile (refreshes the listbox)

 self.BroadcastUpdate
```

```
'''
''' Project.Customize
'''
''' smm 04-99
'''
''' Runs when the Customize.., menu option from the Project
''' menu is selected in the Project window.  It is associated
''' with that menu option's CLICK property.
'''
''' This modified system script prompts the user for a
''' password before opening the Customize window
'''
''' Notes:  The password global variable is set during the
'''         execution of MyProject.StartUp.  Refer to that
'''         script for the password.
'''
''' ----------------------------------------------------------

theProject = av.GetProject
aPass = MsgBox.Password
if (aPass = _thePass) then
  theProject.Customize
else
  MsgBox.Error("Your password is incorrect!","")
end
```

```
'''
''' Project.Exit
'''
''' Runs whenever the user exits from the project with
''' the Exit option from the File menu of any window.
'''
''' This modified system script does not prompt the user
''' for a save when exiting the project.
'''
''' ------------------------------------------------------

theProject = av.GetProject
theProject.Close

av.Quit
```

```
'''
''' Project.Save
'''
''' smm 04-99
'''
''' Runs whenever the user selects a function that initiates
''' a project save.

''' This modified system script will prompt the user
''' for a password.  If it's incorrect, the save will bail.
'''
''' ----------------------------------------------------------

theProject = av.GetProject
theFileName = theProject.GetFileName

aPass = MsgBox.Password
if (aPass = _thePass) then
  if (theFileName = nil) then
    av.Run("Project.SaveAs", nil)
  else
    if (av.Run("Project.CheckForEdits",nil).Not) then
      return nil
    end
    if (theProject.Save) then
      av.ShowMsg("Project saved to '" + theFileName.GetBaseName + "'")
      if (System.GetOS = #SYSTEM_OS_MAC) then
        Script.Make("MacClass.SetDocInfo(SELF, Project)").DoIt(theFileName)
      end
    end
  end
else
  MsgBox.Error("Your password is incorrect!  This project will not be saved.","")
end
```

```
'''
''' Query.Table1
'''
''' smm 05-99
'''
''' Executes when run from the following script:
'''
'''         MyProject.StartUp
'''
''' This script prompts the user to choose the county or counties with which
''' he or she wishes to work.  When chosen, Table1 is queried and the resulting
''' selected set is exported to a table called "qtable".  This table is joined
''' to the PTC Station's attribute table (Attributes of PTC Stations).
'''
''' Notes:  The creation of "qtable" allows the user to define which area he
'''       or she wants to work in.
'''
'''       Since "qtable" is joined to the PTC Stations attribute table
'''       instead of the origin table Table1, the refresh procedure takes
'''       much less time to execute.  This is because refreshes require
'''       tables to be rejoined and any SQL connections associated with the
'''       tables to be requeried.  Since "qtable" is a derived from a
'''       selected set of Table1, it is usually smaller and the rejoin
'''       executes much faster.  Also, it is not created from an SQL query,
'''       and so a requery is not necessary.
'''
''' ------------------------------------------------------------------------

''' Check to see if qtable exists.  This is possible if the project was
''' saved while qtable existed.  If it does exist, throw up an error message
''' and prompt for a password to enter.

errorVTab = av.GetProject.FindDoc("qtable").AsString
if (errorVTab = "qtable") then
  errorVTab = av.GetProject.FindDoc("qtable")
  av.GetProject.RemoveDoc( errorVTab )
end

''' Query by county the Access table to create qtable:

''' Set up a list of counties for the user to choose from

aList = {"ALAMANCE", "ALEXANDER", "ALLEGHANY", "ANSON",
         "ASHE", "AVORY", "BEAUFORT", "BERTIE", "BLADEN",
         "BRUNSWICK", "BUNCOMBE", "BURKE", "CABARRUS",
         "CALDWELL", "CAMDEN", "CARTERET", "CASWELL",
         "CATAWBA", "CHATHAM", "CHEROKEE", "CHOWEN", "CLAY",
         "CLEVELAND", "COLUMBUS", "CRAVEN", "CUMBERLAND",
         "CURRITUCK", "DARE", "DAVIDSON", "DAVIE", "DUPLIN",
         "DURHAM", "EDGECOMBE", "FORSYTH", "FRANKLIN",
         "GASTON", "GATES", "GRAHAM", "GRANVILLE", "GREENE",
         "GUILFORD", "HALIFAX", "HARNETT", "HAYWOOD",
         "HENDERSON", "HERTFORD", "HOKE", "HYDE", "IREDELL",
         "JACKSON", "JOHNSTON", "JONES", "LEE", "LENOIR",
         "LINCOLN", "MACON", "MADISON", "MARTIN", "MCDOWELL",
         "MECKLENBURG", "MITCHELL", "MONTGOMERY", "MOORE",
         "NASH", "NEW HANOVER", "NORTHAMPTON", "ONSLOW",
         "ORANGE", "PAMLICO", "PASQUOTANK", "PENDER", "PERQUIMANS",
         "PERSON", "PITT", "POLK", "RANDOLPH", "RICHMOND",
         "ROBESON", "ROCKINGHAM", "ROWAN", "RUTHERFORD",
         "SAMPSON", "SCOTLAND", "STANLY", "STOKES", "SURRY",
         "SWAIN", "TRANSYLVANIA", "TYRRELL", "UNION", "VANCE",
         "WAKE", "WARREN", "WASHINGTON", "WATAUGA", "WAYNE",
         "WILKES", "WILSON", "YADKIN", "YANCEY", "ALL COUNTIES"}

theChoice = MsgBox.MultiListAsString (aList, "Please select a county", "County Selection")

''' Make sure Table1 is deselected before we start selecting sets from it
```

```
aVTab = av.GetProject.FindDoc("Table1").GetVTab
aBitmap = aVTab.GetSelection
aBitMap.ClearAll

''' Get rid of the Table1-to-PTC Stations
''' attribute table join and rebuild it with only a selected set built from
''' theChoice.

aVtab =  av.GetProject.FindDoc("Attributes of PTC Stations").GetVTab
if ((aVTab.IsBase).Not) then
  aVTab.UnjoinAll
end

'''We're going to query Table1 by CountyID, but first we'll check to see if
'''"ALL COUNTIES" was chosen.  If so, we'll take a different approach.

if (theChoice = nil) then
  MsgBox.Info("NO SELECTION", "")
else
  for each c in theChoice
    aCounty = c.AsString

''' If "ALL COUNTIES" was selected, then we'll select all of the records in
''' Table1.

    if (aCounty = "ALL COUNTIES") then
      theTable = av.GetProject.FindDoc("Table1")
      theTable.GetVTab.GetSelection.SetAll
      theTable.GetVTab.UpdateSelection
      av.GetProject.SetModified(true)

''' Now display the startup banner

      f="d:\seasonal\images\splash.gif".AsFileName
      MsgBox.Banner (f, 5, "")

''' If a selection other than "ALL COUNTIES" was made, then select
''' each record in Table1 by their coresponding county ID.

    else

''' Grab the index number of aCounty from the list above, which happens
''' to be the County-ID in Table1

      theChoice = aList.FindByValue (aCounty)

''' Display the startup banner

      f="d:\seasonal\images\splash.gif".AsFileName
      MsgBox.Banner (f, 5, "")

''' Selecting Table1 by theChoice

      aVTab = av.GetProject.FindDoc("Table1").GetVTab
      aBitmap = aVTab.GetSelection
      theExpression = "([CountyID] = "+theChoice.AsString+")"
      aVTab.Query( theExpression, aBitmap, #VTAB_SELTYPE_OR)
      aVTab.UpdateSelection
    end
  end
end

'''Create a VTab from the selected records of Table1 to join to PTC Stations
'''attribute table

theVTab = av.GetProject.FindDoc("Table1").GetVTab
theVTab.Export ( "qtable".AsFileName, dBASE, TRUE )

''' Now create a VTab from qtable.dbf and set the name to "qtable"
```

```
theFileName = "qtable.dbf".asFileName
theVTab = VTab.Make (theFileName, false, false)
theTable = Table.Make(theVTab)
theName = theTable.SetName ("qtable")

"' And join qtable to Attributes of PTC Stations

aVtab =  av.GetProject.FindDoc("Attributes of PTC Stations").GetVTab
aFromVTab = av.GetProject.FindDoc("qtable").GetVTab
aToField = aVTab.FindField("CovSysID")
aFromField = aFromVTab.FindField("CovSysID")
aVTab.Join (aToField, aFromVTab, aFromField)
```

```
'''
''' View.BackToView1
'''
''' smm 12-98
'''
''' Script runs when the Back to Seasonal Group View button
''' (labeled 'B') is clicked in the View1 (locator map) window
''' interface.
'''
''' When executed, the script returns the user to the SGAA View
''' at the same extent from when he or she entered the Locator
''' Map view.
'''
''' ------------------------------------------------------------
theDoc = av.GetProject.FindDoc("NCSU-ITRE Seasonal Group Assignment Application")

'''Close the Locator Map view
theDoc = av.GetActiveDoc
theDoc.GetWin.Close
```

```
'''
''' View.ExtentRedraw
'''
''' smm 12-98
'''
''' Runs when the Change SGAA View Extent button located on the
''' Locator Map's View interface is clicked.
'''
''' Returns the user to the SGAA View at the extent specified by
''' the user:  The user can resize and/or reposition the rectangle
''' graphic located inside the view.  When this script is executed,
''' the origin and size of the rectangle are stored as variables
''' and the extent of the SGAA view is set to those values.  This
''' creates the effect of an "interactive" locator map.
'''
''' ----------------------------------------------------------------

''' Get the dimensions of the rectangle graphic inside
''' the locator map view and set them to variables
theView= av.FindDoc("Locator Map")
theView.GetGraphics.SelectAll
theGr= theView.GetGraphics.ReturnExtent
theOrigin= theGr.ReturnOrigin
theSize= theGr.ReturnSize
theOriginX= theOrigin.GetX
theOriginY= theOrigin.GetY
theSizeX= theSize.GetX
theSizeY= theSize.GetY

''' Now make the SGAA View active
theDoc= av.FindDoc("NCSU-ITRE Seasonal Group Assignment Application")
if (not(theDoc.GetWin.IsOpen)) then
theDoc.GetWin.Open
end

''' Bring the doc to the front
if (theDoc = nil) then
  av.GetProject.GetWin.Open
elseif (theDoc.GetWin.IsOpen) then
  theDoc.GetWin.Open
end

''' Change the mapextent values to those of the rectangle graphic.
theDisplay= theDoc.GetDisplay
aRect = Rect.Make(theOriginX@theOriginY, theSizeX@theSizeY)
theDisplay.ZoomToRect(aRect)

'''Close the Locator Map view
theDoc = av.FindDoc("Locator Map")
theDoc.GetWin.Close
```

```
'''
''' View.LocatorMap
'''
''' smm 12-98
'''
''' Runs when the Locator Map button in the SGAA View interface is clicked.
'''
''' Makes the locator map view active, which consists of a statewide county
''' coverage and a rectangle graphic that reflects the SGAA View extent.
'''
''' ------------------------------------------------------------------------

'''Capturing the view extent characteristics and setting them to variables
theView= av.FindDoc("NCSU-ITRE Seasonal Group Assignment Application")
d= theView.GetDisplay
theRect= d.ReturnVisExtent
theOrigin= theRect.ReturnOrigin
theSize= theRect.ReturnSize
theOriginX= theOrigin.GetX
theOriginY= theOrigin.GetY
theSizeX= theSize.GetX
theSizeY= theSize.GetY

''' Find the locator map document in the project
theDoc = av.GetProject.FindDoc("Locator Map")
if (not(theDoc.GetWin.IsOpen)) then
  theDoc.GetWin.Open
end

''' Bring the doc to the front
if (theDoc = nil) then
  av.GetProject.GetWin.Open
elseif (theDoc.GetWin.IsOpen) then
  theDoc.GetWin.Open
end

''' Selecting old rectangle graphic
theDoc = av.GetActiveDoc
theDoc.GetGraphics.SelectAll

''' Deleting old rectangle graphic
theView = av.GetActiveDoc
theTheme = theView.GetEditableTheme
if (theView.GetGraphics.HasSelected) then
  av.GetProject.SetModified(true)
end
if (theTheme = nil) then
  theView.GetGraphics.ClearSelected
else
  theTheme.GetFtab.BeginTransaction
  theTheme.ClearSelected
  theTheme.GetFTab.EndTransaction
end

''' Draw the locator box to the extent captured from the SGAA View in the
''' variables above

theView= av.FindDoc("Locator Map")
aRect = Rect.Make(theOriginX@theOriginY, theSizeX@theSizeY)
gr = GraphicShape.Make(aRect)
theView.GetGraphics.UnselectAll

''' Select the graphic so it is ready to be resized/repositioned by the user
gr.SetSelected(TRUE)
theView.GetGraphics.Add(gr)
```

SUPPLEMENT B:

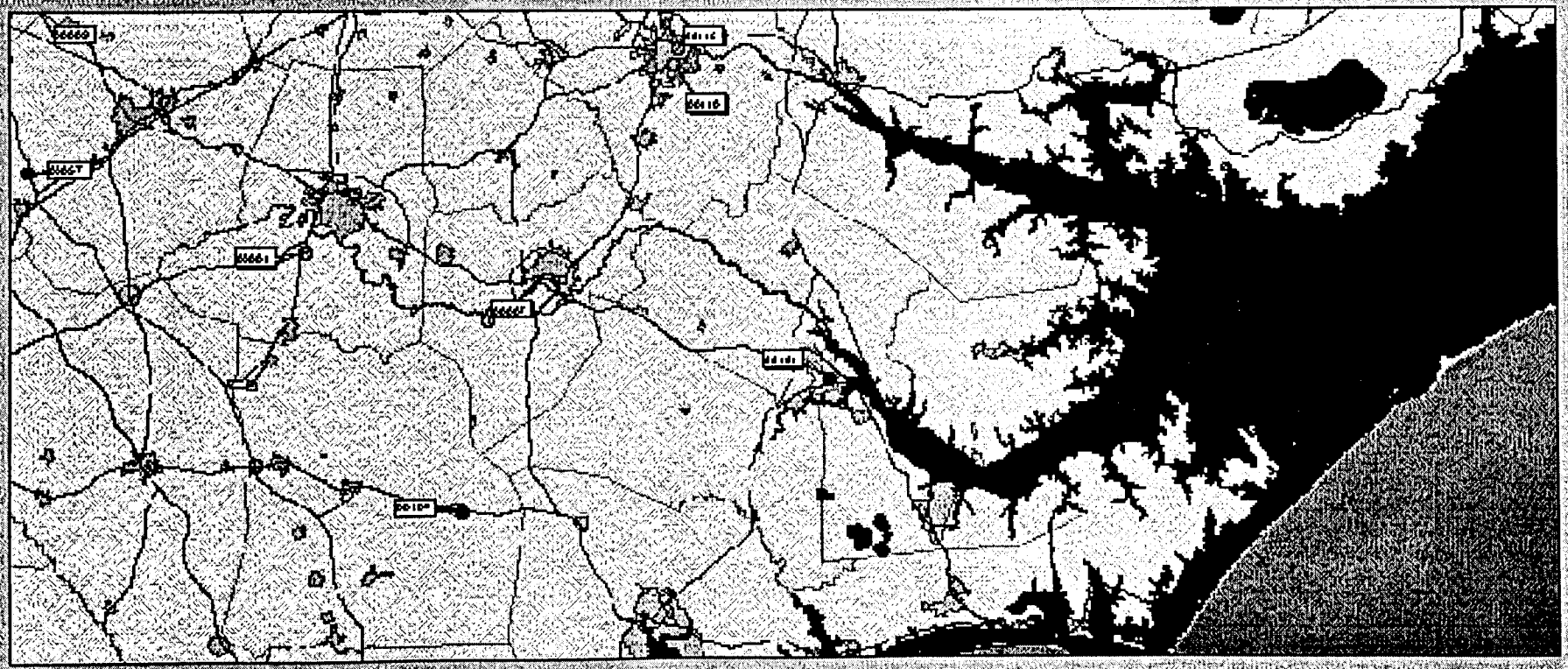
SAS JMP ANALYSIS AND CUSTOMIZED ARCVIEW PROCEDURES TUTORIAL

(Class given for first beta version in May, 1999)

North Carolina Department of Transportation

Statewide Planning Branch

Seasonal Analysis Application



Seasonal Analysis ArcView Program Tutorial

Course Documentation

Part I Introduction and ArcView Review

Part II PTC Site Analysis with ArcView

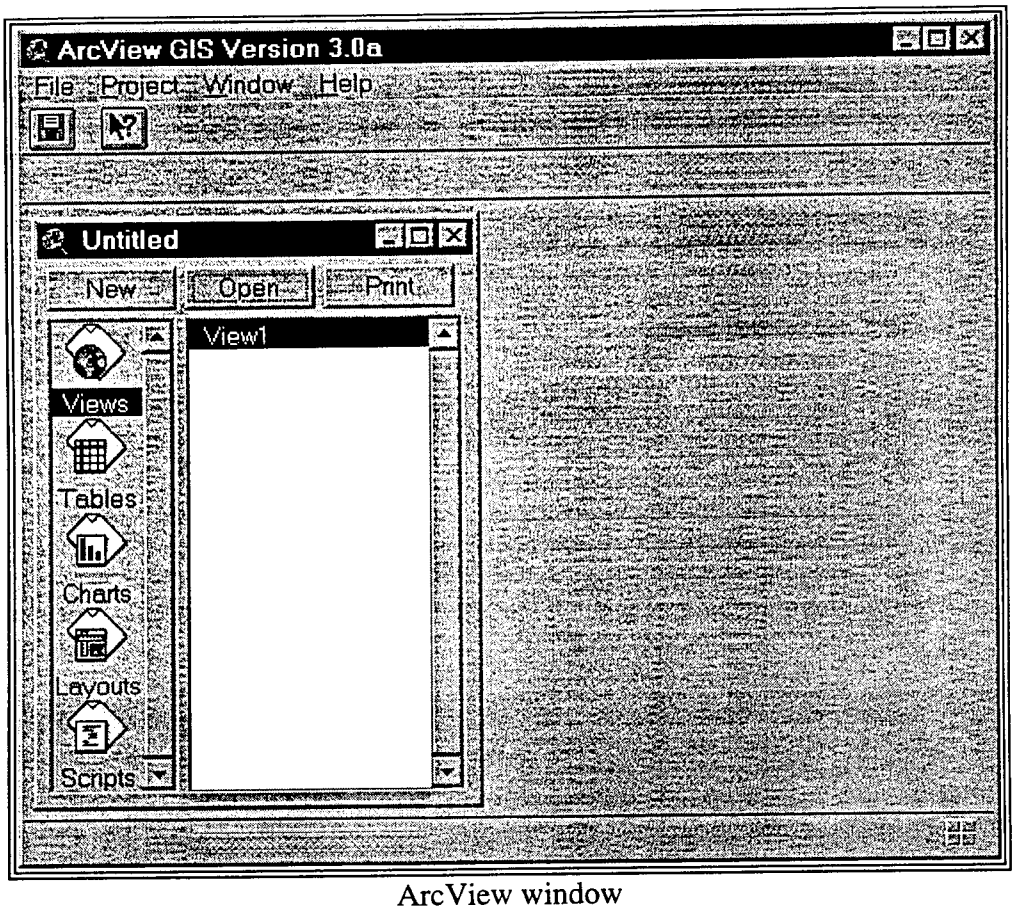
1. ArcView in Review

1.1 What is ArcView?

The software we are about to examine is a tool that brings geographic visualization of data to the desktop. It incorporates a mouse-and-menu interface that allows its user to visualize, query, and analyze data of a spatial nature. The software is available for Microsoft® Windows™, Apple® Macintosh®, and various Unix® platforms.

ArcView can be used with existing ARC/INFO® data sets as well as shapefiles. It can incorporate external data stored in dBASE® or ASCII files. Images, grids, and event data can be used with ArcView.

When using ArcView, you can geocode data using address-matching techniques for display on a map; address logical queries to databases, then display the results; select and display features based on their geographical proximity to each other; create charts and statistical summaries based on your data; and compose and print maps showing the results of your analysis.



ArcView window

1.2 General GIS terms and concepts

Avenue

Avenue is the programming language and customization and development environment of ArcView. Avenue is commonly used to write programs, referred to as scripts, automate repetitive tasks, modify and customize the ArcView interface, develop and distribute new applications, and integrate ArcView with other applications. Avenue is an *Object Oriented* language.

ATR stations

An NCDOT continuous-count traffic station. These sites and their associated data were used to develop the beta version of this application.

attribute (item, field)

Data stored as columns in an attribute or other database table. Items, fields, and attributes are synonymous terms.

coverage (cover)

The data structure within which geographic information and attributes are stored in ARC/INFO. A coverage may contain points, lines, polygons, or other **features**, and a **feature attribute table** associated with each. A typical coverage might be lines (arcs) that represent the individual segments of a road network with an attribute table containing items describing the route number, number of lanes on each segment, etc.

feature

An arc (line), polygon, point, or other symbol representing a geographical entity on Earth. Nodes (the beginning and ending points of an arc), label points, and annotation are other examples of features. A feature has characteristics, or attributes, that are stored as records in an attribute table.

feature (theme) attribute table

Database table comprised of attributes (columns) and records (rows). Each record in a feature attribute table represents a geographic entity (feature) in a coverage. The intersection of a record and attribute is the value of a particular feature for that attribute. Feature attribute tables, as with other tables in ArcView, are dynamic. This means they reflect the current status of the tabular data source they represent, and not the data itself. This characteristic allows theme attribute tables to be edited without changing the source table itself. ArcView's feature attribute tables are in Dbase format and carry the extension *.dbf*.

hotlink

A hotlink gives the user ability to associate each feature in a theme with some action.

project (.apr)

A project is a set of pointers to and descriptions of a collection of components as stored by ArcView. The components can be documents (views, tables, layouts, charts, and script editors), graphic user interfaces (GUI), and scripts.

shapefile

The data structure within which geographical features and their associated attributes are stored in ArcView. Shapefiles can be created in ArcView, or converted from (or to) ARC/INFO coverages.

theme

A theme is a collection of features drawn in a view and is associated with a classifiable legend that allows the features to be symbolized by values from the theme's feature attribute table.

value

The intersection of a record and an item in a database. See **attribute table.**

1.3 Starting ArcView

We'll need to know how to start up the ArcView software before we go any further with our discussion. If we were using the Unix platform, you would need to start ArcView from the Unix prompt in any convenient window by typing its name followed by an ampersand:

```
aurora:/u4/class1 1 % arcview &
```

To start ArcView under Microsoft Windows or on a Macintosh, simply double-click on the appropriate icon.

ArcView Icon

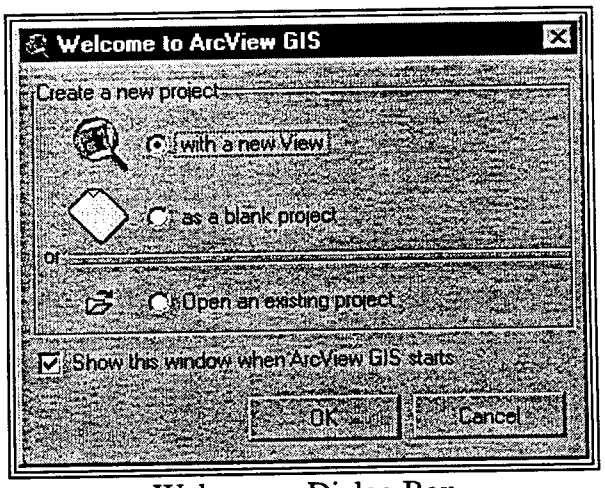
In either case, you'll be presented with the main ArcView window, which will look slightly

different on each platform. The Windows version is the one we've pictured in the ArcView window figure in section 1.1.

2. Introduction to the Seasonal Analysis ArcView Program

2.1 The Project Window and its components

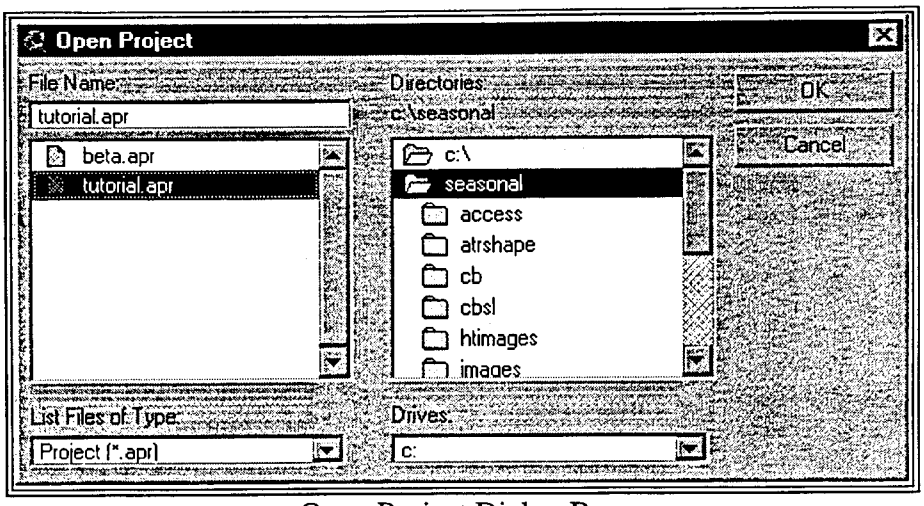
When ArcView 3.1 opens, you will be prompted by a Welcome... dialog box to choose between creating a new project with a view, creating a new blank project, or open an existing project. Choose the latter and click **OK**.



Welcome... Dialog Box

If working in an earlier version of ArcView, or no dialog opens, then wait until the program is loaded and choose **Open Project...** from the **File** menu.

The project is loaded in **c:\seasonal** and is called **tutorial**. Highlight the file *tutorial* under the **seasonal** folder and click **OK**.



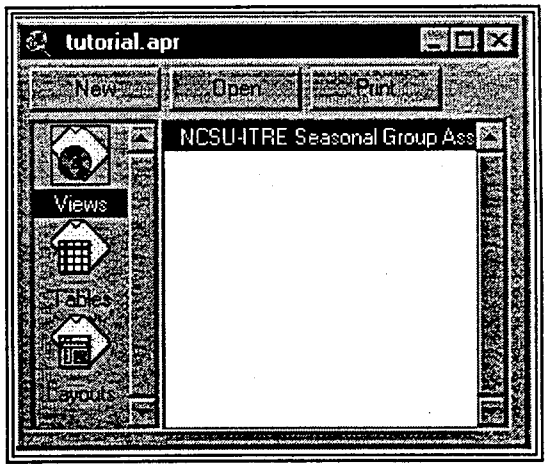
Open Project Dialog Box

Notice the Project Window is now titled **tutorial.apr**

2.2 The Project Window and its components

Recall that the project window usually contains five component classes that are accessible to the user. They are the Views, Tables, Charts, Layouts, and Scripts. The component classes carries a list of each component in the project. For example, if a View is created and named MyView, then MyView would be listed under the Views component class. MyView could then be opened, copied, edited, etc. from the project window.

However, to protect the scripts from inappropriate editing and to simplify the custom project for the user, only three of these components will be accessible; Views, Tables, and Layouts.



Tutorial Project Window

Note that components can be added to the project, but CAN NOT BE SAVED without a password. For example, a layout can be created in the project, but it would be necessary to plot the layout or save it to a graphics file before exiting.

Let's take a look at each component in our project.

The View components

- NCSU-ITRE Seasonal Group Assignment Application -- the digital map display of the PTC and ATR sites. The View also includes reference themes such as primary roads, secondary roads, railroads, surface water, municipality boundaries, counties, and four annotation layer themes for the ATR sites. This theme will heretofore be called *SGAA*.

The Table components

- Attributes of PTC Accepted Stations -- the attribute table associated with the *PTC Accepted Stations* theme. The table is joined to *Table1* for access to the *AcceptFlag* field, which is used to classify the *PTC Accepted Stations* theme.

- Attributes of PTC Stations -- the attribute table associated with the *PTC Stations* theme. The table is joined to *Table1* for access to its fields when building a selected set for editing.

- Table1 -- the Access table as seen in ArcView via an ODBC connection. This table is used as a link between ArcView and Access.

- newfile -- a temporary table created when the PGA Update Dialog Box is opened. *Newfile* acts as an editing platform and link between the dialog box and *Attributes of PTC Stations*. Its contents only contain the selected records of *Attributes of PTC Stations*.

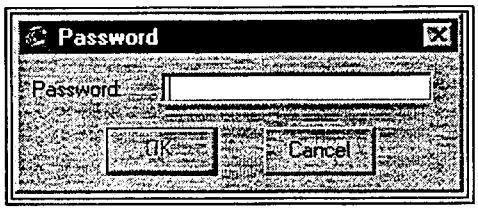
The Layout components

- No layout components exist. Layouts can be created and added to the project, but they must be saved as a graphics file or plotted before exiting the program. A template may be added to the program and saved by the systems administrator or anyone who has editing access.

We will take a closer look at the interfaces for these windows later.

2.3 User Accessibility and the Reset Date

To safeguard the program, only program users with the correct password are allowed access to saving and customizing the interface. If you are prompted with the choice to save and click **Yes**, then you will be asked to enter a password. If the wrong password is given, or if **Cancel** is clicked, the save is disregarded and an error message is given.

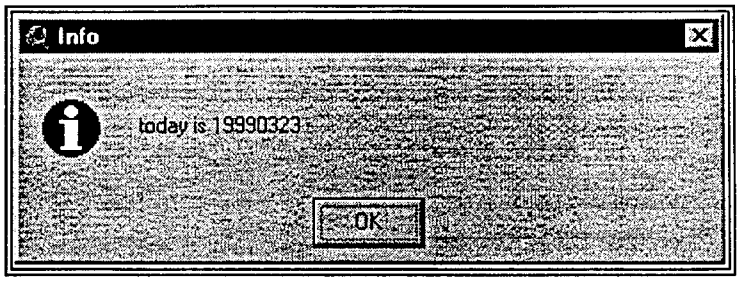


Tutorial Project Window

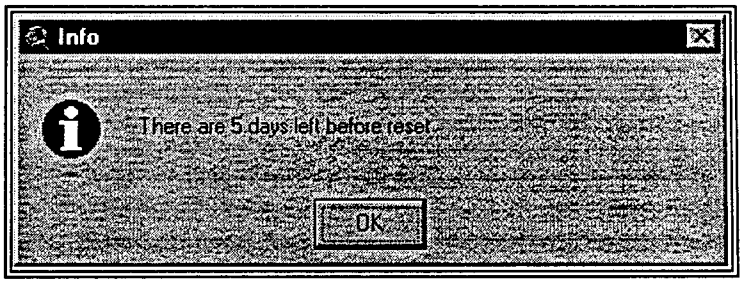
To exit the program without being prompted, choose **Exit** from the **File** menu of any window.

Once a year, the administrator will reset the Access database. On that date and until the administrator has reprogrammed the reset date, access will be denied to anyone who does not have the correct password.

For thirty days prior to the reset date, the user will be informed by two message boxes of the current date and the number of days left until the password is required for access.



Info Message Box with Today's Date



Info Message Box with Countdown to Reset Date

3. The View Window

3.1 The View table of contents & display properties

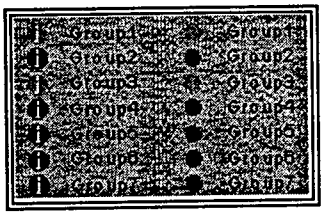
As we mentioned earlier, there are several themes located in the *NCSU-ITRE Seasonal Group Assignment Application* View's table of contents. Most themes are present for spatial reference, but they can also be used for selecting PTC sites by geographic proximity (e.g. select all PTC site features that are within the selected county from the *North Carolina County* theme). There will be a demonstration of this selection method later in the tutorial.

Some themes have display properties that only allow them to be viewed once within a set range of scale. For example, the secondary roads only become visible when the display is at or below a scale of 1:250000.

Let's take a look at each of the themes and their display properties:

- PTC Accepted Stations -- this theme is classified by the *AcceptFlag* field from *Table1*. Only those PTC stations that have an *AcceptFlag* value greater than zero (i.e. a date) will be displayed in this theme. The effect is a circle and crosshairs around those sites that have been "accepted".

- PTC Stations -- a point feature theme that represents the PTC stations statewide. The theme is classified by the *Classify* field, which is a positive or negative number derived from the Pending Group Assignment (or PGA) value of each site. Each PGA value is coded by a unique color in the display. For example, all sites with a PGA value of four (4) are blue, those with a PGA value of five (5) are green, and so on. If the station has been analyzed with SAS JMP, then the *Classify* field is positive and the feature is represented by a solid circle with an "i" stamp. If no analytical data is present for the site, its *Classify* field is negative and the feature is represented by a plain solid circle.



PTC Stations
Classified

- ATR Stations -- statewide coverage of the ATR stations. The stations are classified by their *ATRGroup* field values. These values presently range between one and seven (1-7) representing their best fit curve group.

- Primary Road -- statewide coverage of North Carolina's primary roads. This includes all interstate, US, and major NC routes.

- Secondary Road -- statewide coverage of North Carolina's secondary roads. The display properties are set so that the coverage is only visible at a scale greater than 1:250000.

- Railroad -- statewide coverage of North Carolina's railroad tracks. The display properties are set so that the coverage is only visible at a scale greater than 1:250000.

- Surface Water -- major surface waters of North Carolina, including major rivers, lakes, and sounds.

- Municipality -- municipal boundaries for North Carolina's cities and towns. The display properties are set so that the coverage is only visible at a scale greater than 1:50000.

- North Carolina County -- coverage of all 100 North Carolina Counties.

- Annotation Layer 1 -- includes labeling for the ATR stations by their unique station number. This annotation layer displays between scales 1:100000 and 1:300000.

- Annotation Layer 2 -- includes labeling for the ATR stations by their unique station number. This annotation layer displays between scales 1:300001 and 1:550000.

- Annotation Layer 3 -- includes labeling for the ATR stations by their unique station number. This annotation layer displays between scales 1:550001 and 1:850000.

- Annotation Layer 4 -- includes labeling for the ATR stations by their unique station number. This annotation layer displays at a scale less than 1:850000.

- Road Annotation Layer -- includes labels for most of North Carolina's major highways. The theme displays between the scales 1:550000 and 1:850000.

Exercise

The display properties can be changed during the ArcView session for any of the themes. Let's take a look at how the display properties work for the *Municipality* theme.

- Check the box beside the *Municipality* theme name so that it will draw. Do any changes take place inside the View? Why or why not?

- Open the **Theme Properties** dialog box: Choose **Properties...** from the **Theme** menu or click the *Theme Properties* button in the button bar.

- At what range of scales does the theme display?

Now we're going to change the scale properties so that *Municipality* displays between the scales of 1:25000 and 1:100000.

- Set the minimum scale to 1:25000 by typing **25000** in the box provided next to *Minimum Scale: 1:*
- Set the maximum scale to 1:100000 by typing **100001** in the box provided next to *Maximum Scale: 1:* and Click "OK".

Note that the maximum scale is set to a value of 100000 + 1. This is required in order to draw the theme at a scale of 1:100000. In mathematical terms, our theme will display if the following equation is true:

```
        1:25000 >= display extent  < 1:100001
or
        1:25000 >= display extent <= 1:100000
```

Let's test our new display properties:

- Locate the editable box next to **Scale 1:** in the tool bar. Type in **50000** and press **Return** on your keyboard. Do the municipality boundaries draw? Why or why not?

By default, many of these themes are not set visible. This saves time when the display redraws after zooming, panning, or opening the view. Remember to check the box next to the theme's name to draw the features.

3.2 Moving around in the SGAA View -- the zoom buttons and interactive locator map

Let's try the various methods of moving around the view. First, take a look at the group of seven zoom buttons in the button bar.

The first button in the collection allows a quick zoom to the extent of all themes in a view; the second zooms to the selected theme. The third button zooms to the extent of only those features that have been selected in a particular theme. The fourth button zooms into the center of the view; the fifth has the opposite efffect. The sixth button (second from right) brings you back to the previous extent you were viewing. You can use this option to retrace your last five steps.

We'll discuss the last button in a moment, but first let's take a look at the zoom tools.

The tool bar also includes some tools that allow you to specify which part of the map is in view; these tools are pictured below:

The first two are *tools* that allow you to zoom in and out, but these work differently than their counterparts (i.e. the zoom buttons); select the **zoom-in** tool (far left) to see how it works. First, notice that the shape of your cursor changes to reflect the tool you've chosen. Now click once on any part of your map; a zoom-in is executed, using the point upon which you clicked as the new center of the screen. If you drag a rectangle shape with your cursor, the map zooms to the extent of that rectangle. Try this now. The **zoom-out** tool (middle) has similar functionality. The third button is the **Pan** tool; you can grab your screen and push it in any direction to change your view. Try out this button now. Zoom to the extent of the "PTC Stations" theme before continuing.

 Locator Map button

The *Locator Map* button calls up a customized view with a N.C. County coverage and a rectangle graphic drawn to the current extent of the *SGAA* window.

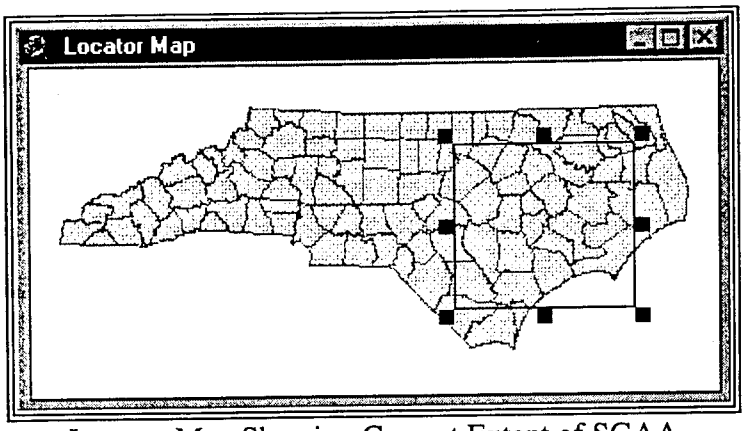
Click the *Locator Map* button. Notice the black rectangle indicates the current extent of the previous view and that the button bar and tool bar has changed. Notice also that there are two new customized buttons; *Change SGAA Extent* and *Back to SGAA*.

 Back to SGAA button

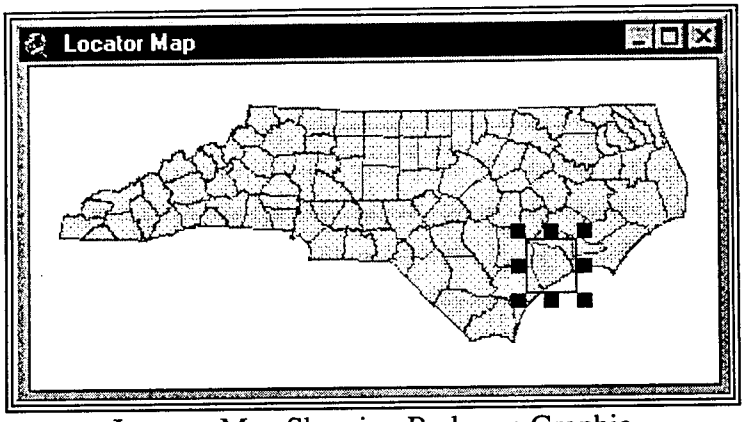
The *Back to SGAA button* will take us back to that view without any changes to the extent. The locator map will disappear and the SGAA view will become active.

 Change SGAA Extent

It is possible to change the extent of the SGAA view from the locator map by resizing and/or repositioning the rectangle graphic. For example, if we were interested in the area around Jackson County, we could resize and reposition the rectangle so that it just fits the extent of that county:



Locator Map Showing Current Extent of SGAA



Locator Map Showing Redrawn Graphic

To reposition the box, use the *Pointer* tool to grab the rectangle along the border and drag it to the new location. To resize the box, place the cursor over any of the black boxes. The cursor will change to indicate the direction in which the rectangle can be resized. Click and drag the rectangle to the desired position.

Once the rectangle is redefined, the user can click on the *Change SGAA Extent* button to dismiss the Locator Map and make active the *SGAA* View redrawn to the specified extent.

Exercise

- Use the locator map to change the current extent of the SGAA View to show only Johnston County and the immediate surrounding area.

3.3 Methods of Selection

Recall that in ArcView we can contain as many selected sets as there are themes in the project. Also recall that we can only have one selected set at any given time. For the next few sections, we're going to revisit several selection methods and apply them to our project.

3.3.1 Selection by pointing or drawing

The easiest way to select features is by clicking or dragging with the *Select Feature* tool from the toolbar.

 Select Feature tool

Make sure your *PTC Stations* theme is active, select this tool, and point to a PTC station. The point becomes highlighted. Now drag a rectangle over a number of stations and watch them all become selected. Features can be added or deleted from the selected set by holding the **Shift** key down and clicking on a feature.

 Clear Selected Feature button

Remember you can clear selected features in a view using the *Clear Selected Feature* button.

3.3.2 Selection with Find

The *Find* button allows an informal search of the values of all the attributes in a particular theme or themes.

 Find button

To try out this feature, let's make the **ATR Stations** theme active. Click on the *Find* button, then type the value you wish to find among the attributes of this theme -- try the text string **A5982.** When you press the **OK** button, the display will shift to place the first feature found in your selected theme in the center of your display. If you are zoomed out to far, use the *Zoom In* button or tool.

3.3.3 Theme on theme selection

You can select the features of an active theme using the features of another theme with the select-by-theme method. For example, let's say we wanted to select all of the PTC stations that are within a quarter mile (roughly 400 meters) of a primary road.

Exercise

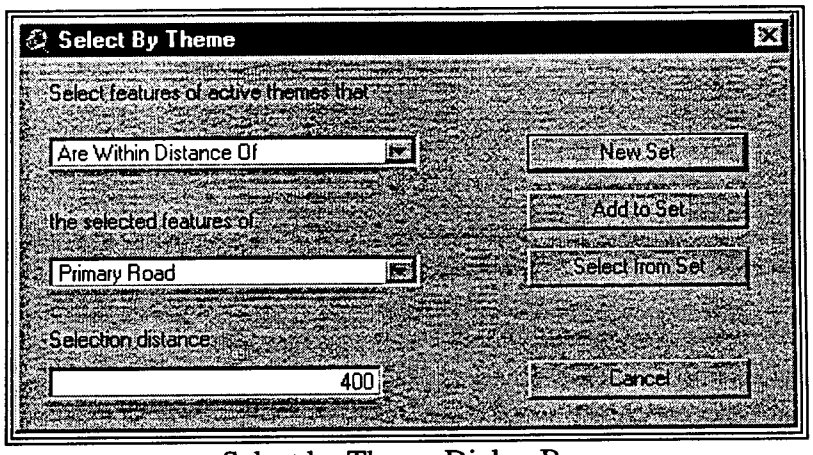
- We'll be working with two themes, so it's important to know which is the selector theme and which is the theme we will be selecting. Since we are interested in selecting the features in *PTC Stations*, we should make that theme active. *Primary Road* will be our selector theme.

- Now choose **Select By Theme...** from the **Theme** menu to summon the **Select by Theme** dialog box.

- Next, choose the theme *Primary Road* as the selector theme from the drop-down list under the text **the selected features of**.

- Now choose **Are Within Distance Of** from the drop-down list located under the text **Select features of active themes that**.

- Type in **400** in the space provided below **Selection distance:** The finished dialog box should look like the figure below:



Select by Theme Dialog Box

- Take a minute to read through the dialog box. Does what you have make since?

- Finally, click on *New Set*. Once the selected set is complete, open the attribute table (remember the *Open Theme Table* button). How many PTC stations are selected? Close the attribute table.

- You'll find that only the selected features of the selector theme will be used to select; use this knowledge to select the PTC stations that are located in Wayne County.

3.3.4 Selection by Logical Query

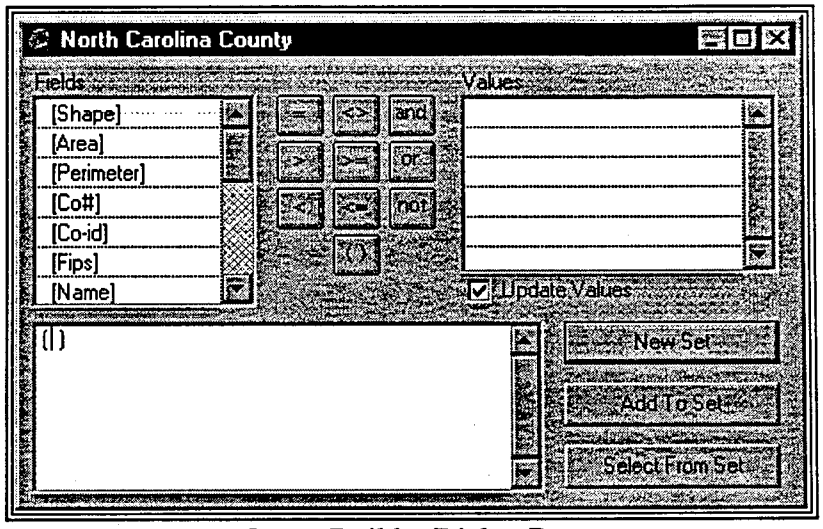
 Query Builder button

Still another way to select features from a theme is to use the **Query Builder** to specify the desired selected set.

Exercise

Let's locate all of the counties in the Department of Transportation's Division 5. The DOT Division values for each county is located in the field *Dot-division.*

- First, make the *North Carolina County* theme active.
- Next, we need to summon the **Query Builder** dialog box, either by selecting **Query** from the **Theme** menu, or by clicking the *Query Builder* button. The Query builder dialog box will appear, showing the available attributes in the data source (not just the one symbolized in the theme).



Query Builder Dialog Box

- Create a logical expression in the dialog box that will select the counties whose ***Dot-division* = 5**
- Choose *New Set* and dismiss the dialog box. How many counties are in Division 5?

3.4 The Accept PGA button

In a moment, we're going to discuss how to change and accept the Pending Group Assignment (PGA) for each PTC station in the **PGA Update Dialog Box**. But first, let's take a look at how we can accept the current PGA without having to open the dialog box.

 Accept PGA button

By clicking the *Accept PGA* button, the user is indicating that he or she has observed the station and is satisfied with the current PGA assignment.

To "accept" a station, the feature has to be selected. This can be done with any of the above methods, but most likely will be selected by using the *Select Feature* tool. Once a feature is selected, the Accept button can be clicked to update the *AcceptFlag* field in *Table1* and subsequently the Access table with the current date as a numerical value. The View will automatically refresh itself to show the change.

Multiple stations can be accepted by selecting all of the features in question and then clicking the *Accept* button.

Exercise

- Let's go ahead and "accept" the selected set from the exercise above. Now clear all selected features.
- Take a look at the view. What changes have taken place?

---

4. The PGA Update Dialog Box

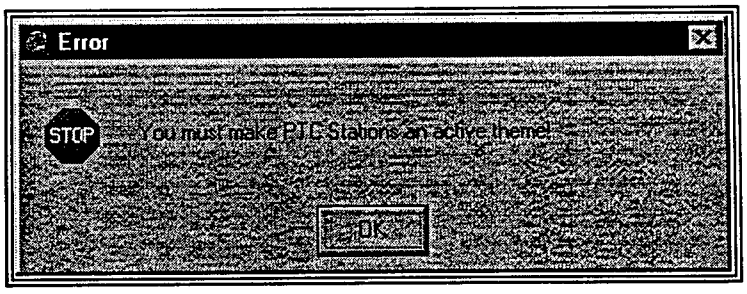
The **PGA Update Dialog Box** is the editing platform for changing field values in the Access table. Once displayed, the user will have access to several buttons that can be used to populate the *PGA* and *AcceptFlag* fields. When changes are applied from the dialog box, they are written to a temporary intermediate table called *newfile* which in turn writes the edits to *Table1*. The *newfile* table is dismissed and, if it is opened, the SGAA view refreshes to reflect any changes that may have occured. While the dialog box is opened, no access to the ArcView interface is allowed. This is to ensure that *newfile* and any other table is not tampered with in that such misuse would lead to program errors.

4.1 Opening the PGA Update Dialog Box

When the **PGA Update Dialog Box** is opened, a temporary "translator" file is created that contains only the selected records from the *Attributes of PTC Stations* table. The records from *newfile* are used to populate the dialog box's listbox. Thus, the listbox reflects the selected set from the *PTC Station*'s attribute table.

 Open PGA Update Dialog Box button

This button can be accessed from either the SGAA view or the table window interface. If the dialog box is launched from the SGAA view, the PTC Stations theme must be active. If it is launched from the table window interface, the Attributes of PTC Stations must be active. An error message will display if the wrong theme or table is active when the button is clicked.



When the dialog box is opened, the user will have access to change the PGA for any record in the listbox. In addition, the user can either "accept" or "unaccept" any PGA field value he or she wishes.

Exercise

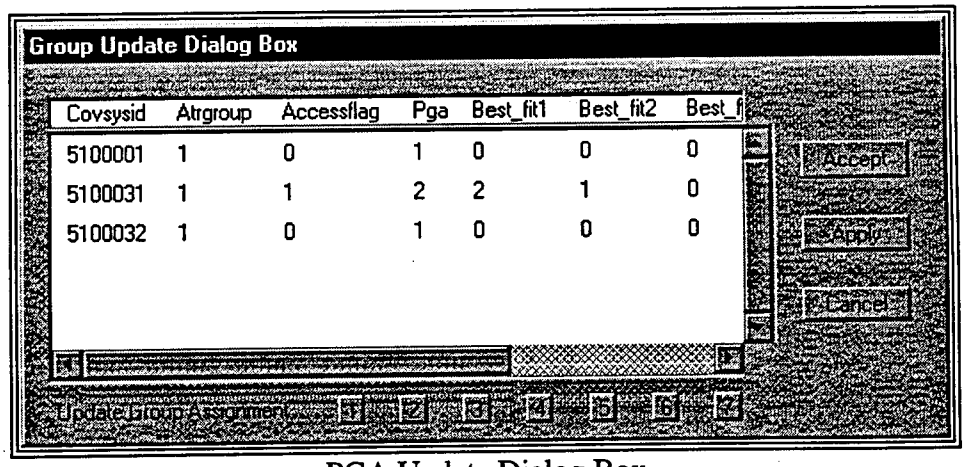
Let's select a few stations and change their *PGA* and *AcceptFlag* values.

- First, select all of the PTC stations that *Are Completely Within* Jones County. Zoom to the extent of the selected themes.
- Clearselect the set and examine the features' colors. Take a note on the Group values for each site (i.e. their colors). Also notice that the site furthest to the left has been "accepted".
- Now select the three stations again, this time spatially with the *Select Feature* tool.
- Open the PGA Update Dialog Box, there should be three records listed.

4.2 Components of the PGA Update Dialog Box

4.2.1 Items and values in the listbox

The most noticable feature of the dialog box is its listbox; the white area that resembles a table. The values inside the listbox are derived from the selected records in the *Attributes of PTC Stations* table. Take a moment and observe the attributes and values in the listbox.



PGA Update Dialog Box

The *Covsysid* is the unique identifier and is our link to both *newfile* and the theme attribute table.

Also present are the fields *PGA*, *Best_fit1*, *Best_fit2*, and *Best_fit3*. Another field we are interested in is *AcceptFlag*. You should notice that the first record has already been accepted, and was done so on March 17, 1999.

Further examination will reveal that not all of the stations have JMP SAS data available for them, i.e. they do not have an "i" stamp in the view. Only those stations that have an *Accessflag* equal to one (1) have such data. If this is the case, then the *PGA* is equal to *Best_fit1*. Otherwise, the *Atrgroup* value is assigned to the *PGA* field.

4.2.2 The Update PGA Dialog Box buttons

In addition to the listbox, there are several buttons presented by the dialog box that allows edits to be made to the *PGA* and *AcceptFlag* fields. These edits are written to the *newfile* table and rewritten to the Access table when an *Apply* is given.

Exercise

- Select the first record, Covsysid 5100001, and notice the buttons are now accessible.

Let's take a moment to look at each button on the dialog box:

- Update Group Assignment buttons (1-7) -- these buttons update the *PGA* field value with the value depicted by the buttons' labels (i.e. the button labeled "7" will calculate the *PGA* field to equal seven (7) for the selected record).

- Accept/UnAccept button -- this button toggles between *Accept* or *UnAccept* depending on the value of the *AcceptFlag* for the selected record. If the value is zero (0), then the station's PGA has not been accepted and the *Accept* button is accessible. If the value is greater than zero (0), i.e. there is a date-stamp, then the PGA value has been accepted and the user has the option to "unaccept" the station with the *UnAccept* button. Unaccepting a station's PGA will change the value of the *AcceptFlag* to zero (0).

- Apply -- updates the Access table with the edits made in the dialog box. The dialog box is dismissed and the SGAA view and *PTC Stations* theme attribute tables refresh.

- Cancel -- dismisses the dialog box without editing the Access table.

Exercise

- Select the second record in the listbox, Covsysid 5100031. Click on the *1* update group assignment

button. Now *Accept* the site. Scroll over to see today's date in the *AcceptFlag* field.

- Double-click the last record. Notice the PGA is now zero (0) for that record. Double-clicking updates the *PGA* field with the value from *Best_fit1*. Let's change it to two (2) and *Accept* the PGA.

4.2.3 Applying the changes to the Access table

When clicked, the *Apply* button runs a script that cursors through each record in *newfile* and writes the values to the Access table via SQL. The *newfile* and *Table1* records are joined by the unique values of *Covsysid*. The *newfile* table is dismissed, along with the dialog box, and the PTC attribute tables are refreshed to reflect changes.

Exercise

- Click *Apply* to write the edit changes to the Access table. When the dialog box dismisses, clearselect your PTC Stations. What changes have taken place?

Now let's check to see that the write has taken place in Access:

- Exit out of ArcView from the **File** menu.
- Open up Access from the start menu. We'll want to open the existing database named **C:\seasonal\access\sax** so highlight that choice and click **OK**.
- You'll see that the database consists of four tables. Highlight the table *tblSeasonalFactorStations* and click the *Open* button. *Table1* in ArcView is derived from this table. Scroll down and find station 5100031. Notice that the PGA is equal to one (1) and the *AcceptFlag* field is stamped with today's date. Take time to observe the new values for the other station we changed; 5100032.
- Exit from Access and open the **tutorial** project in ArcView. Notice that the SGAA View is drawn to the same extent as it was when we last exited ArcView.

---

5. SAS JMP Output Visuals

Part of the SAS JMP analysis required the creation of several GIF images for each station. The images were then pasted into a template in the Paint Shop Pro® program. These images can be launched from the ArcView SGAA View interface. In a moment we're going to see how this is done.

First, let's take a look at some other images available at the custom view window's GUI.

5.1 The ATR Groups menu and ATR Graphs image button

An image of graphs for all seven groups can be displayed by clicking the *ATR Graphs* button located on the SGAA View button bar.

 ATR Graphs button

Click this button now. Click the "X" gadget in the upper right-hand corner to dismiss the image. If you wish to keep the image window open, then you can either minimize it with the "_" gadget or just click inside the ArcView window to promote it to the front of your screen. To reaccess the image window, locate the icon at the bottom of your screen and click once.

Another group of images are accessible from the custom GUI's **ATR Groups** menu. Each option under the menu launches a graph depicting a group's best-fit curve. There are seven options presently available under the menu, one for each of the groups one (1) through seven (7).

Click on the **ATR Groups** menu button and choose **Group 1**. The graph for that image will display on the right side of your screen. Multiple images can be launched by returning to the menu and selecting a different choice.

5.2 The customized PTC Station SAS JMP analysis hotlinks

Perhaps one of the more useful images would be the SAS visual outputs we discussed earlier. Each station that has undergone the SAS JMP process has an individual GIF format image. These files reside in the *images* subdirectory in the *C:\seasonal* workspace.

A customized hotlink has been created to access these files from the SGAA View. The hotlink script runs whenever the *SAS Visual Hotlink* tool is clicked and a feature is selected.

 SAS Visual Hotlink button

Click on this button, then on one of the stations with the bottom of the cursor. If an image is present for the site, it will display in an image window at the right of your screen. Remember that multiple image windows can be opened at any given time, so make sure you close them when you are done.

Amendment 1

2.2 The Project Window and its components

The Table components

- Attributes of PTC Accepted Stations -- this table no longer exists within the ArcView Seasonal Analysis project.

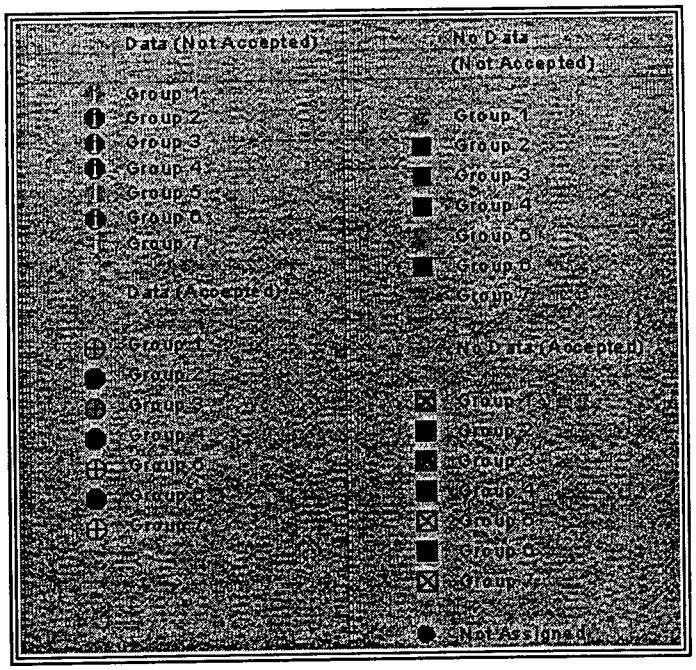
Amendment 2

3.1 The View table of contents & display properties

- PTC Accepted Stations -- this theme no longer exists as part of the beta.apr project. Instead, **PTC Stations** is now classified into four (4) classes.

- PTC Stations -- a point feature theme that represents the PTC stations statewide. The theme is classified by the *Classify* field, which is a positive or negative number derived from the Pending Group Assignment (or PGA) value of each site. Each PGA value is coded by a unique color in the display. For example, all sites with a PGA value of four (4) are blue, those with a PGA value of five (5) are green, and so on. If the station has been analyzed with SAS JMP, then the *Classify* field is positive and the feature is represented by a solid circle with an "i" stamp. If no analytical data is present for the site, its *Classify* field is negative and the feature is represented by a solid square.

  If the station has been 'accepted', the positive or negative value is multiplied by 100. For example, if the station has been analyzed and accepted, and has a PGA value of 4, then its Classify value is 400.



PTC Stations Classified

Notice that there is a fifth classification; Not Assigned. If a station is 'Not Assigned', it has a PGA value of zero (0).

Amendment 3

4.2 Components of the PGA Update Dialog Box

4.2.2 The Update PGA Dialog Box buttons

Let's take a moment to look at each button on the dialog box:

- Update Group Assignment buttons (1-7) -- these buttons update the *PGA* field value with the value depicted by the buttons' labels (i.e. the button labeled "7" will calculate the *PGA* field to equal seven (7) for the selected record). In addition, the *AcceptFlag* field updates to reflect the current date.

- Accept/UnAccept button -- this button toggles between *Accept* or *UnAccept* depending on the value of the *AcceptFlag* for the selected record. If the value is zero (0), then the station's PGA has not been accepted and the *Accept* button is accessible. If the value is greater than zero (0), i.e. there is a date-stamp, then the PGA value has been accepted and the user has the option to "unaccept" the station with the *UnAccept* button. Unaccepting a station's PGA will change the value of the *AcceptFlag* to zero (0).

- Apply -- updates the Access table with the edits made in the dialog box. The dialog box is dismissed and the SGAA view and *PTC Stations* theme attribute tables refresh.

- Cancel -- dismisses the dialog box without editing the Access table.

In addition to the buttons, the user can double-click on a record to update its *PGA* field with the value of *Best_Fit1*. The *AcceptFlag* field is also updated to reflect the current date.

Prototype Demonstration of a Geographic Information System Application for the Seasonal Analysis of Traffic Data, Development of Seasonal Factors and Seasonal Adjustment of Roadways

Institute for Transportation and Education

North Carolina State University

GIS Program

Seasonal Project -- SAS JMP Tutorial

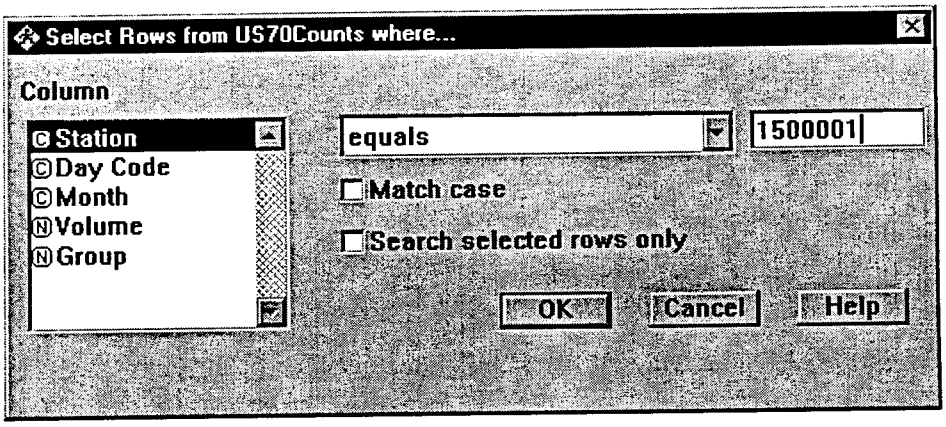
The following guide was used in training the DOT Survey Unit with the necessary skills in SAS JMP needed to produce analytical visual outputs and *Best_Fit* values for each station.

1. Open the data tables that you will use. When in JMP, look in Gis (C):, they are found in the folder c:\seasonal\.

```
File Open:  Station009.jmp
File Open:  SeasoNdx.jmp
File Open:  US70Counts.jmp
```

2. In the data table *US70Counts.jmp*, select the rows for the Station# you will be working on.

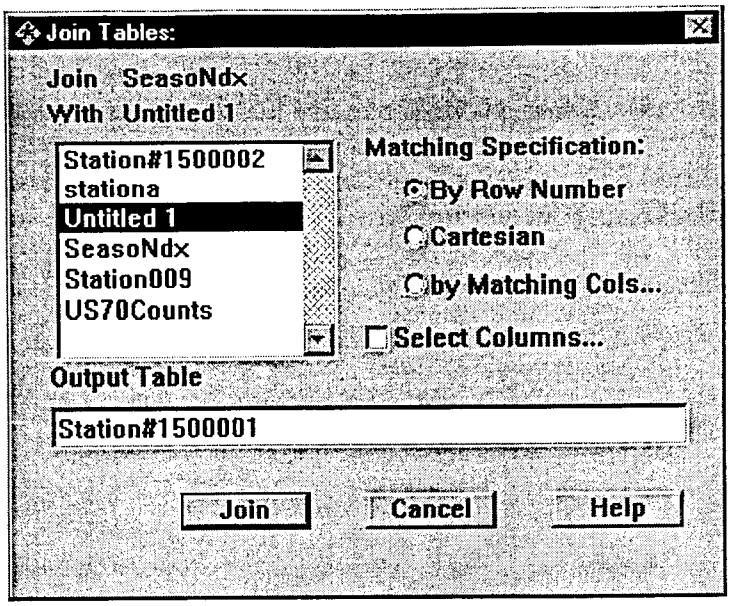
```
Rows Select Where:  Station, equals, station#, OK
```



Select Rows were *Station* = 5100001

3. Only a few columns from *US70Counts.jmp* and *SeasoNdx.jmp* will be needed for statistical graphs. The columns **Station** and **Volume** from *US70Counts.jmp* will be combined with the table *SeasoNdx.jmp* for analyzation.

```
Highlight columns STATION and VOLUME:  Hold down control and click
with your left-most mouse button, simultaneously, in the column header
Tables Subset.

You will get an Untitled table with 84 rows and 2 columns.

Window SeasoNdx
Tables Join:  Untitled #1, By row number, Station#---, Join
```



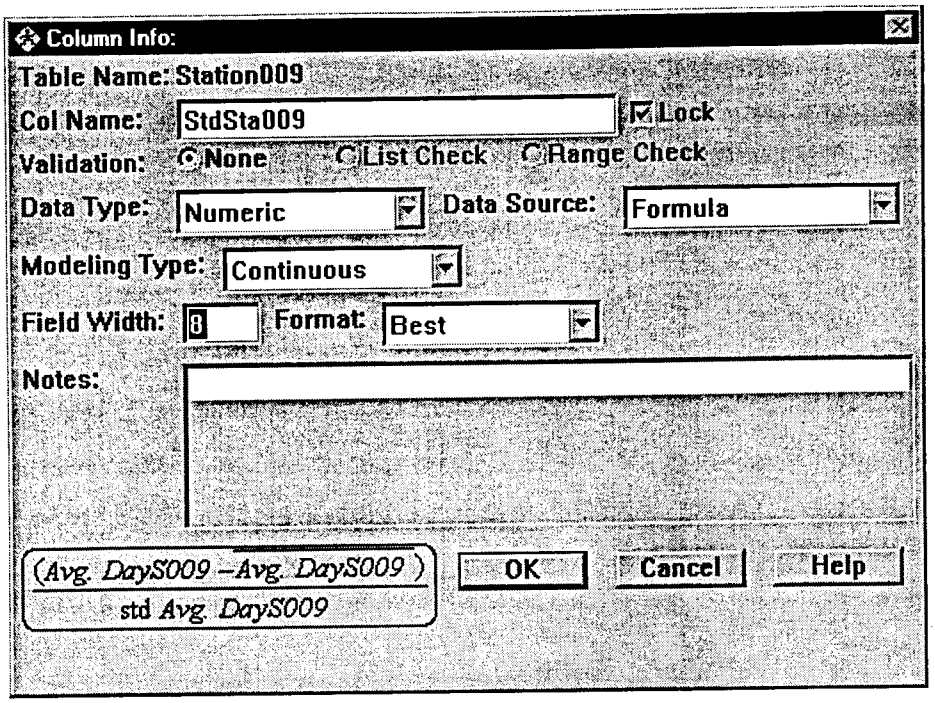
Joining Tables

You should get a table with 84 rows and 25 columns.

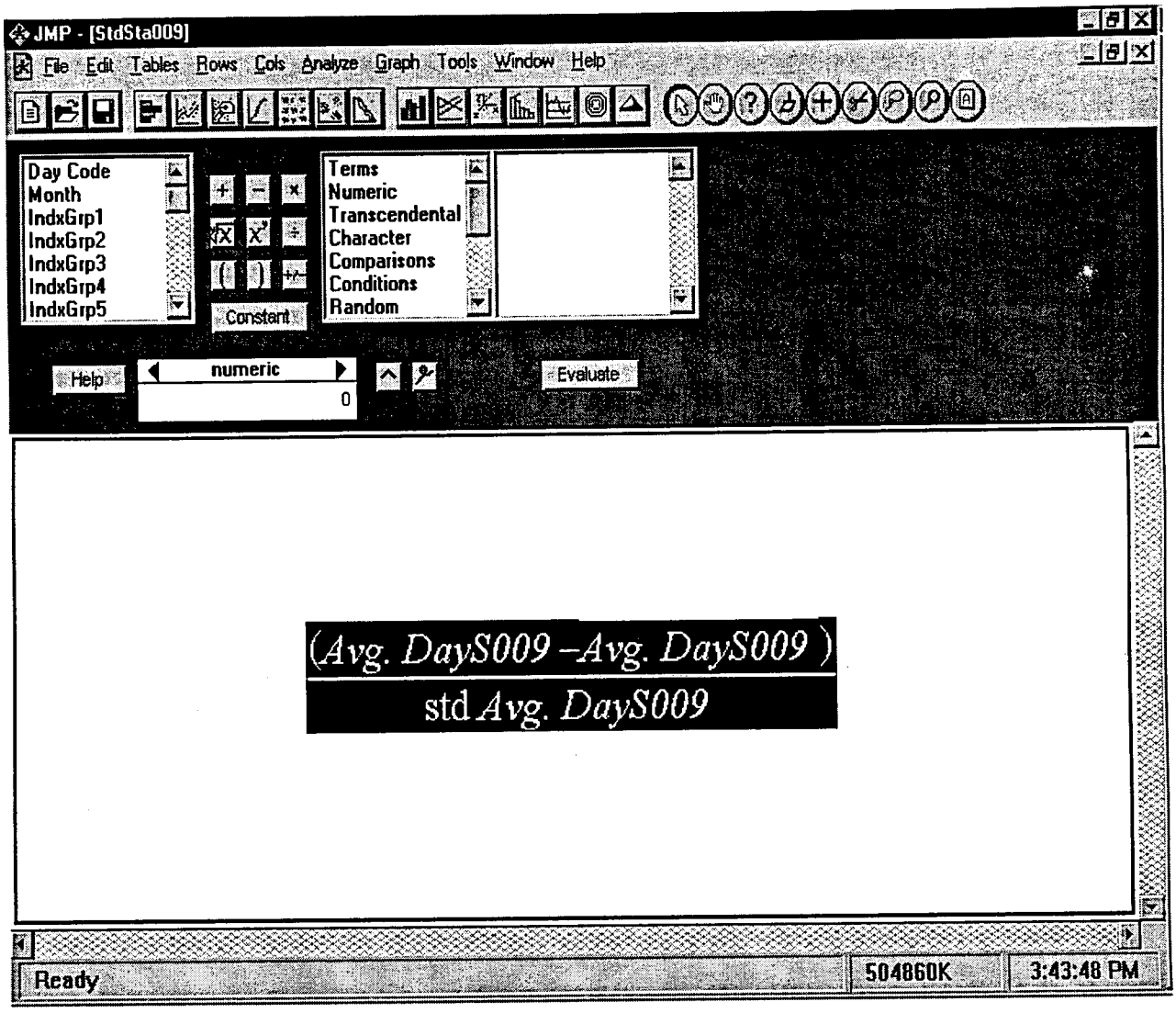
4. Save the combined table as Station#---.jmp in the folder c:\seasonal. Change the name of the column **Volume** by double clicking in the column header. It will need to be changed to Avg.DayS#--- to match the formula used.

5. A formula is needed to obtain the y-axis information for the seasonal pattern of each station.

```
Window Station009

       Double click on the column header StdSta009
       (it is in the middle, the 18th column)

       Click on the formula in the lower left-hand corner

       Click on the fraction bar in the formula to highlight
       all of it
```



Obtaining Formula



Formula

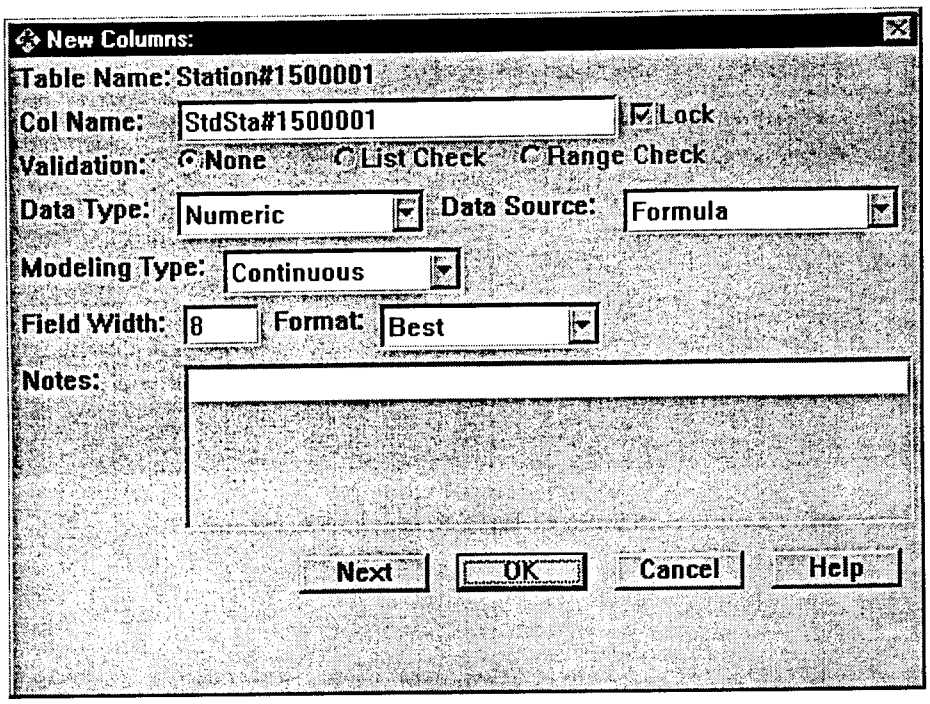
Choose **Edit** Copy

Close the formula window [X]

6. Now that we have obtained the formula we must apply it to our data.

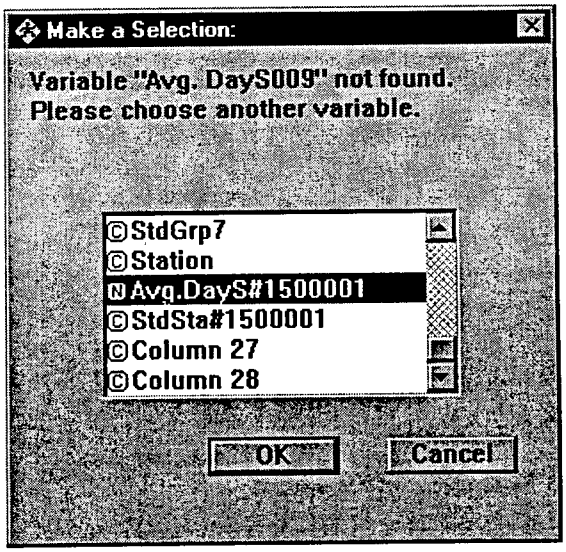
Window Station#

Cols New Column: StdSta#---, Formula, **OK**



New Column

Edit Paste: Avg.DayS#---, **OK**



Column

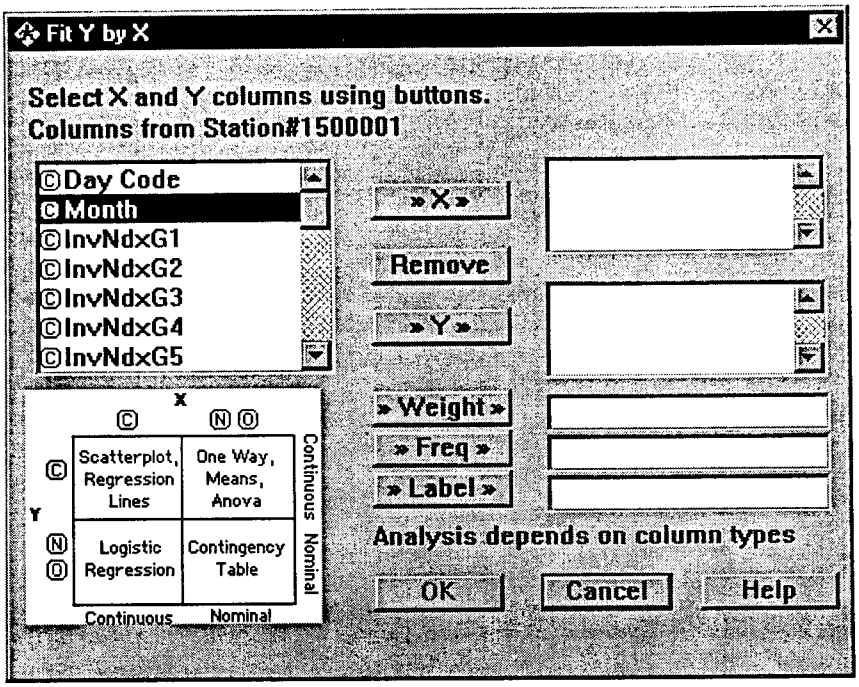
Close the formula window [X]

File Save

7. The first analyzation is a fit Y by X. This will plot StdSta# against Month.

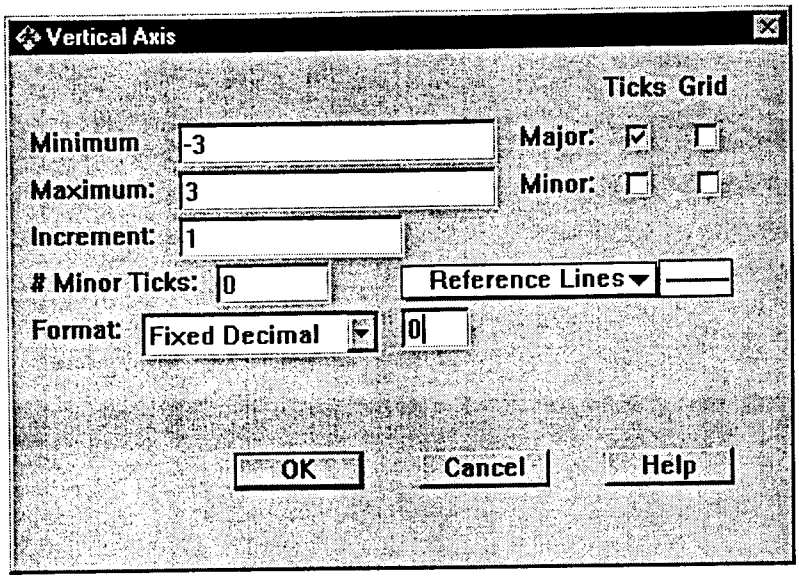
Window Station#---

Analyze Fit Y by X: Month, X, StdSta#---, Y, **OK**



Fit Y by X Dialog Box

Double click on a number on the Y axis:
-3, tab, 3, tab, 1, tab, 0, tab, 0, OK



Vertical Axis Properties

```
Double click on a number on the X axis:
6, tab, 13, tab, 1, tab, 0, tab, 0, OK

Next click on the fitting button in the lower left hand corner:
Fit Spline, .1
```

project home *continue*

Date last modified: March 30, 1999

Seasonal Project -- SAS JMP Tutorial

The next session of the tutorial was designed to capture the station visual outputs to be used by ArcView's hotlink tool. The original tutorial captured four images, each placed into a custom template designed in Paint Shop Pro. The original tutorial follows:

1. The image is now ready to get packaged for Paint Shop Pro.

   Enlarge the image to full screen:

    Enlarge tool

   Click on the enlarge tool

    Scissors tool

   Click on the scissors tool

   Drag the scissors tool into the header (StdSta#--- By Month)

   Highlight the image: Alt-Click

   Edit Copy

2. Open Paint Shop Pro (Start, Programs, Paint Shop Pro, Paint Shop Pro 4).

   File Open: *Image14.gif*

    Crop tool

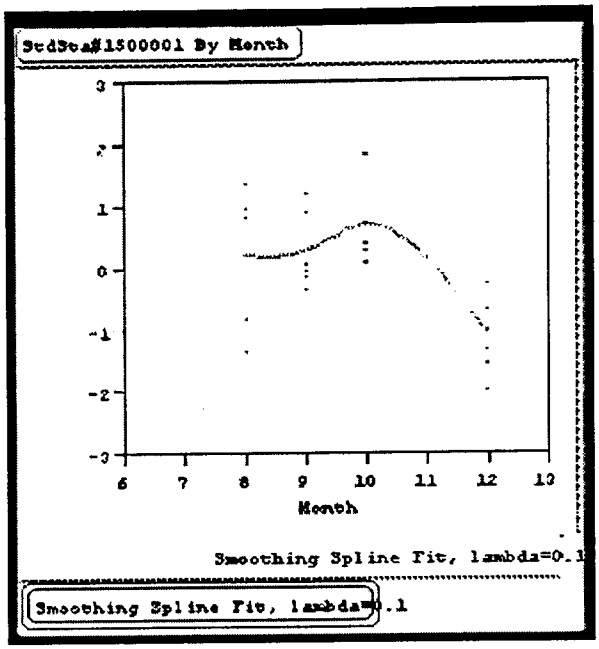
   Click on the crop tool to define a geometric selection

   Click in the upper left corner and drag down to the opposite corner

   Edit Paste Into Selection

   Set Width Pixels: 1000, tab, OK

   File Save as: station#---a



Station5100001a.gif

File Close

 Minimizer button

Click the minimizer button to get back to JMP.

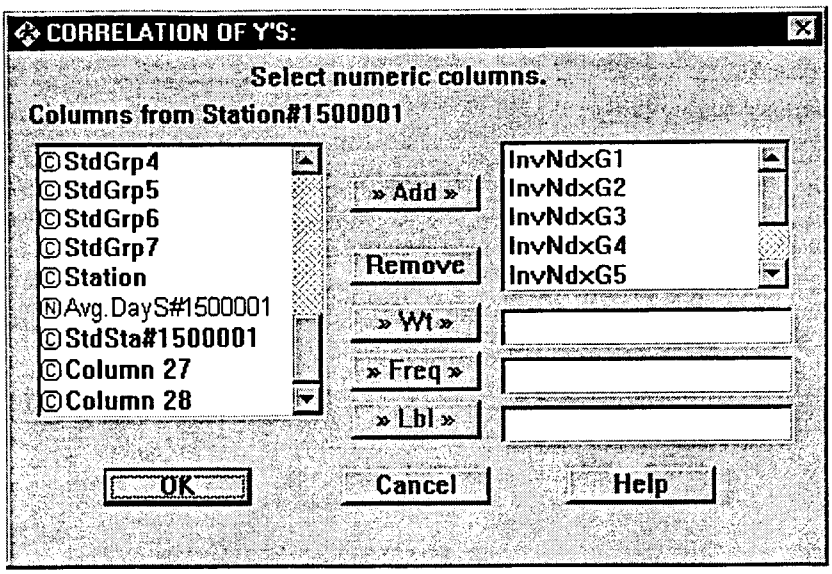
3. The second analyzation is the correlation of Y's.

Window Station#---

Analyze Correlation of Y's

Click on InvNdxG1, hold down shift, click InvNdxG7, **Add**

Scroll down and click on Avg.DayS#---, **Add**



Correlation of Y

Click on the little arrow on the bottom of the screen

Click Correlations-Pairwise

Click on the scissors tool

Drag the scissors into the header (Pairwise Correlations)

Highlight the image: Alt-Click

Edit Copy

4. Go to Paint Shop Pro.

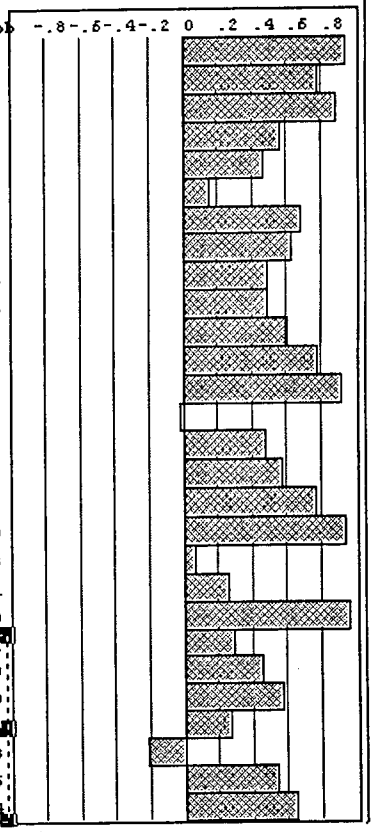
File Open: *Image15.gif*

Edit Paste As New Image OK

Click the crop tool;
You want to define a box around the last seven lines including all four columns but not the picture

Pairwise Correlations

Variable	by Variable	Correlation	Count	Signif Prob
InvNdxG2	InvNdxG1	0.9387	60	0.0000
InvNdxG3	InvNdxG1	0.7754	60	0.0000
InvNdxG3	InvNdxG2	0.8834	60	0.0000
InvNdxG4	InvNdxG1	0.5602	60	0.0000
InvNdxG4	InvNdxG2	0.4595	60	0.0002
InvNdxG4	InvNdxG3	0.1544	60	0.2389
InvNdxG5	InvNdxG1	0.6715	60	0.0000
InvNdxG5	InvNdxG2	0.6152	60	0.0000
InvNdxG5	InvNdxG3	0.4832	60	0.0001
InvNdxG5	InvNdxG4	0.4862	60	0.0001
InvNdxG6	InvNdxG1	0.6058	45	0.0000
InvNdxG6	InvNdxG2	0.7826	45	0.0000
InvNdxG6	InvNdxG3	0.9180	45	0.0000
InvNdxG6	InvNdxG4	-0.0179	45	0.9073
InvNdxG6	InvNdxG5	0.4659	45	0.0013
InvNdxG7	InvNdxG1	0.5781	55	0.0000
InvNdxG7	InvNdxG2	0.7648	55	0.0000
InvNdxG7	InvNdxG3	0.9342	55	0.0000
InvNdxG7	InvNdxG4	0.0573	55	0.6778
InvNdxG7	InvNdxG5	0.2533	55	0.0621
InvNdxG7	InvNdxG6	0.9535	40	0.0000
Avg.DaySJ1500001	InvNdxG1	0.2916	17	0.2562
Avg.DaySJ1500001	InvNdxG2	0.4500	17	0.0699
Avg.DaySJ1500001	InvNdxG3	0.5782	17	0.0150
Avg.DaySJ1500001	InvNdxG4	0.2566	17	0.3005
Avg.DaySJ1500001	InvNdxG5	-0.2152	17	0.4068
Avg.DaySJ1500001	InvNdxG6	0.5374	8	0.1696
Avg.DaySJ1500001	InvNdxG7	0.6424	17	0.0054



Edit Copy

Click on Image15

With the crop tool, create a box from the upper left corner to the bottom right corner

Edit Paste Into Selection

File Save as: station#---b

Avg.daySJ1500002	InvNdxG1	0.4026	24	0.0511
Avg.daySJ1500002	InvNdxG2	0.5645	24	0.0027
Avg.daySJ1500002	InvNdxG3	0.5868	24	0.0025
Avg.daySJ1500002	InvNdxG4	0.2579	24	0.2057
Avg.daySJ1500002	InvNdxG5	-0.1675	24	0.4337
Avg.daySJ1500002	InvNdxG6	-0.0049	10	0.9892
Avg.daySJ1500002	InvNdxG7	0.5555	24	0.0048

Station5100001b Image

File Close

Click the minimizer button on the pairwise correlation image,
you will need it later

5. The third analyzation is in the form of a scatterplot matrix.

In JMP, click on the little arrow on the bottom of the screen

Click Scatterplot Matrix

Drag the scissors tool into the header (Scatterplot Matrix)

Highlight the image: Alt-Click

Edit Copy

6. It's now time to go back to Paint Shop Pro.

File Open: *Image16.gif*

Edit Paste As New Image OK

Click the crop tool;
You'll want to define a box around the last line
going across all the columns

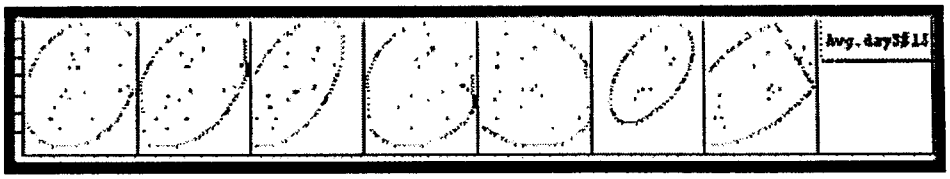
Edit Copy

Click on Image16

Create a box from the upper left corner to the bottom right corner

Edit Paste Into Selection

File Save as: station#---c



Station5100001c Image

File Close

Click the minimizer button on the scatterplot matrix,
you will need it later

7. Go back to JMP to get the image from Y:X.

Window Station#--- Y by X

Drag the scissors tool into the header (StdSta#--- By Month)

Hightlight the image: Alt-Click

Edit Copy

8. Go back to Paint Shop Pro.

File Open: *Sasblank.gif*

Click on the crop tool

Click in the square from the upper left corner to the bottom right corn

Edit Paste Into Selection

Set Width Pixels: 1000, tab, OK

Click in the box right below the square and crop the whole box

Click on your maximize button on the *correlation of Y's* image

Highlight the same region you did last time

Edit Copy

Click in the box you cropped in the sasblank image

Edit Paste Into Selection

Crop the last box in your sasblank image

Click on your maximize button on scatterplot matrix

Highlight the same region as the first time

Edit Copy

Click in the box in the sasblank image

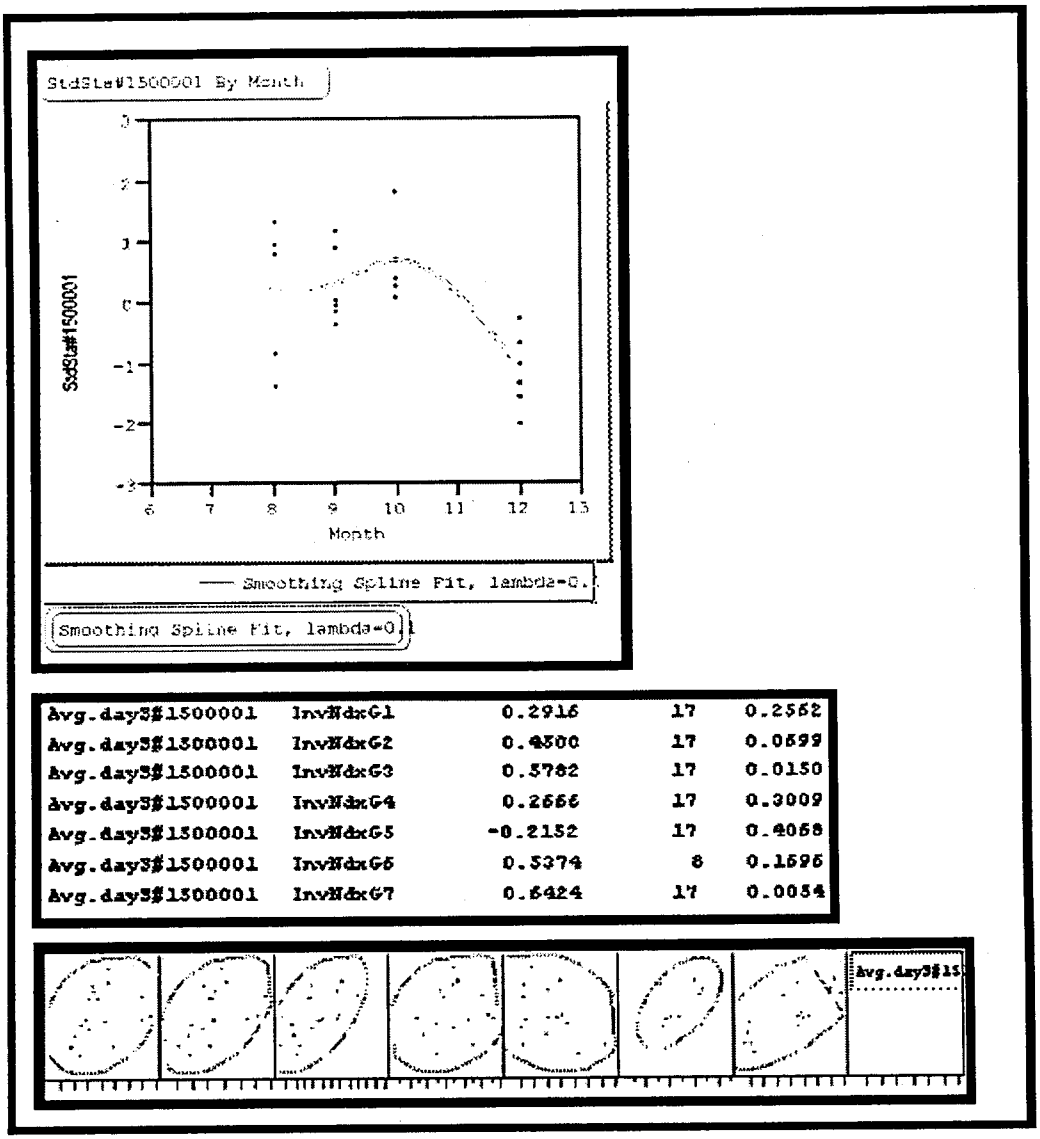
Edit Paste Into Selection

File Save as: station---

File Exit

```
You are done!
```

9. The finished product:



SAS Analysis Visual Output.gif

back *project home*

Date last modified: March 30, 1999