9. Classic Analysis

Introduction

Classic Analysis manipulates, manages, and analyzes data. It acts as a statistical toolbox providing many ways to transform data and perform statistical Classic Analysis. Data can be selected, sorted, listed, or manipulated with a series of commands, functions, and operators. Available statistics include frequencies, means, and more advanced processes (i.e., Kaplan-Meier Survival Classic Analysis and Logistic Regression). Classic Analysis can be accessed by clicking **Classic** from the Epi Info™ main window or by selecting **Tools > Analyze Data>Classic** . It reads data files created in Form Designer and other types of databases (e.g., MS Access, MS Excel, SQL Server, and ASCII). Classic Analysis can also produce graphs to present graphic representations of data.
Classic Analysis provides access to existing data directly or through forms created in Form Designer. It reads data from files and tables created in Epi Info 7, Microsoft Access, Microsoft Excel, SQL Server, and ASCII.

Navigate the Classic Analysis Workspace

The Classic Analysis module contains four areas: the Classic Analysis Command Tree, Program Editor, Classic Analysis Output window, and the Message Area.

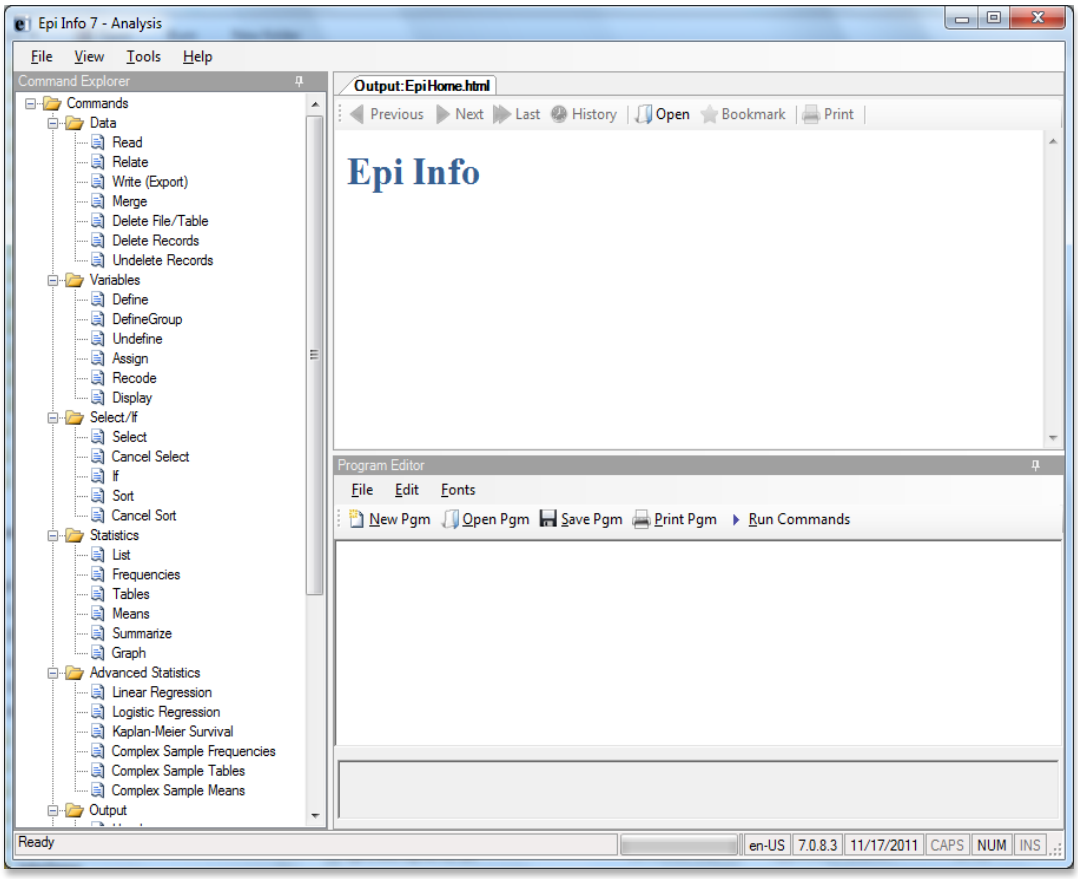


Figure 9.1: Classic Analysis Workspace

1. The Command Explorer contains a list of available commands separated into folders by command type. Commands are generated, edited, and executed. Selecting a command opens the corresponding dialog box for that command, function, or statistic to run.

2. The Program Editor displays the commands and code created using the Classic Analysis Command Tree. Commands can also be typed directly into the Program Editor. Programs or .PGM files written in Classic Analysis can be stored in the current .PRJ or as text files. Saved programs can be run against new data, or shared with others.

3. The Output window acts as a browser and displays information generated from commands run in Program Editor. The buttons allow you to navigate through program scripts that run and display in the output screen.

4. The Message window alerts you if any problems occur with any executed commands.

How to Manage Data

Use the READ Command

To analyze data, it must be read or imported into Classic Analysis. The READ command allows you to select a project and/or data table to run statistics. The READ command is used almost every time you open Classic Analysis.

Warning! Never manipulate the Form table in a software application outside of Epi Info 7. The database may become corrupt and render the form unusable.

1. From the Classic Analysis Command Tree Data folder, click **Read.** The READ dialog box opens.

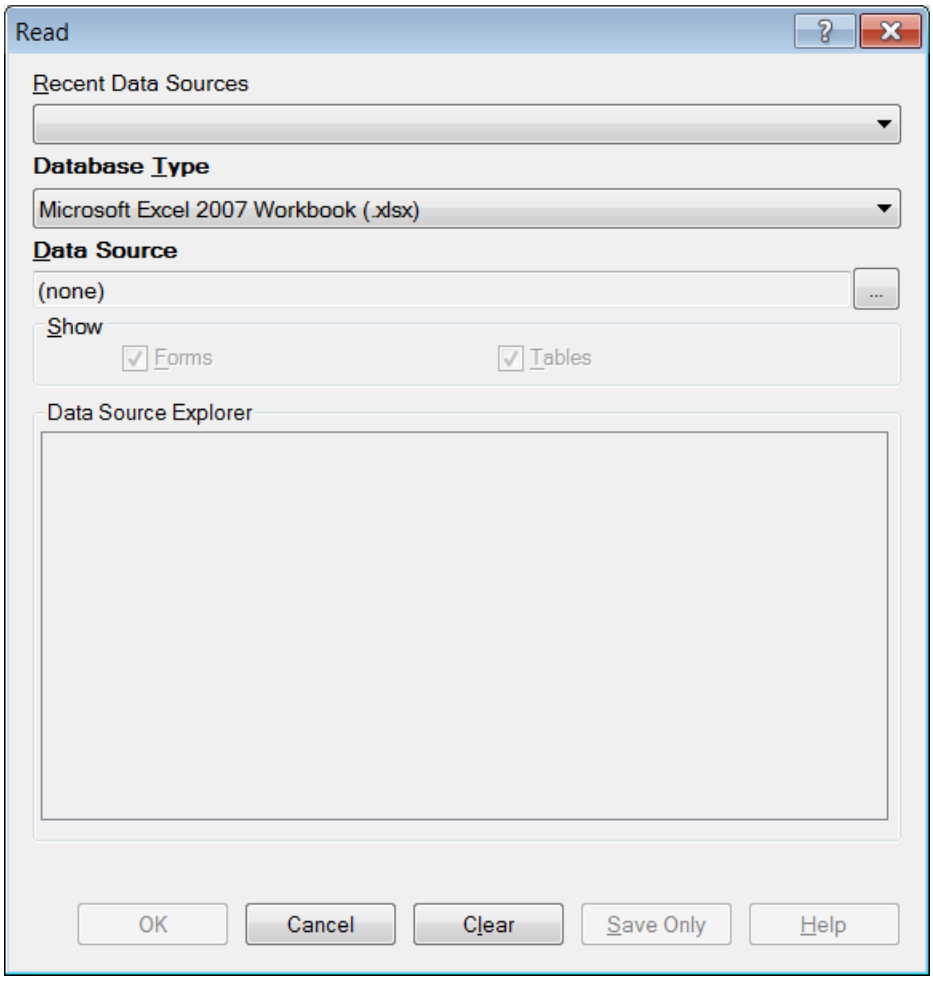


Figure 9.2: Read Window

- The **Database Type** field indicates the database file to be loaded (.PRJ, .MDB, .XLS). Specify the data format for the data to be read. Unless otherwise specified, output files created from these data are stored in this destination folder.

- The **Data Source** field indicates the file location/path. If you use an SQL Server database, the Data Source field will require server and database names.

2. The **Recent Data Sources** field provides a list of recently-accessed databases in Classic Analysis or Visual Dashboard. By selecting one of the data sources available on the list, you do not need to provide Database Type or Data Source information. Only data tables or forms will need to be selected again. If you are reading an Epi Info 7 project file from the Data Source section, select the **Forms** checkbox to view available forms in the project, or select the **Tables** checkbox to view all project tables.

3. From the Forms section, select a **form** in your project. This reads the **data table** associated with your form.

4. Click **OK**. The READ command is saved in the Program Editor and run simultaneously. The current form file location, Record Count, and Date appear in the Classic Analysis Output window; the READ command appears in the Program Editor.

    - Click **Save Only** to save the command in the Program Editor. The command does not run and the Record Count is not displayed.

Read a Microsoft Excel File

---

1. From the Classic Analysis Command Tree, click **Read.** The READ dialog box opens.
2. From the Data Formats drop-down list, select either **MS Excel 97-2003 Workbook** or **MS Excel 2007 Workbook.**
3. In the Data Source field, click on the **Browse** button to browse and select the **workbook (.xls or .xlsx)** to import into Classic Analysis.
4. Click **Open.**
   - First Row Contains Field Names is checked by default. If the first row of the spreadsheet does not contain field names, deselect the **checkbox**.
5. Click **OK**
6. Select a **Worksheet** from the list provided in the Data Source Explorer.
7. Click **OK**. The Record Count and file information appear in the Classic Analysis Output window.

Read a Microsoft Access File

---

1. From the Classic Analysis Command Tree, click **Read.** The READ dialog box opens.
2. From the Data Formats drop-down list, select either **MS Access 2002-2003 (.mdb)** or **MS Access 2007(.accdb).**
3. In the Data Source field, click on the **Browse** button to browse and select the **file** to import into Classic Analysis.
4. Click **OK.**
5. Select a **data table** from the list provided in the Data Source Explorer.
6. Click **OK**. The Record Count and file information appear in the Classic Analysis Output window.

Read an ASCII Text File

---

1. From the Classic Analysis Command Tree, click **Read**. The READ dialog box opens.
2. From the Data Formats drop-down list, select **Flat ASCII File**.
3. In the Data Source field, click the **Browse** button to browse.
4. Click the **Browse** button again. Select **directory** of the file to import into Classic Analysis.
5. Click **OK**.
6. Select the **file** from the list provided in the Data Source Explorer
7. Click **OK.**

Note: There are two forms of text files. Since both have only one table per file, you do not have to specify a table. Both put the data for one record on a single line. The difference is in how the fields are indicated.

Read an SQL Server Database

---

1. From the Classic Analysis Command Tree, click **Read**. The READ dialog box opens.
2. From the Data Formats drop-down list, select **Microsoft SQL Server Database**.
3. In the Data Source field, click the **Browse** button to browse.

4. When the Connect to SQL Server Database dialog opens, specify the server name and database name to connect and import into Classic Analysis.
5. Click **OK**.
6. From the list provided in the Data Source Explorer, select a **data table**.
7. Click **OK**. The Record Count and file information appear in the Classic Analysis Output window.

Use Related Forms

To use RELATE, the READ command must open at least one form/table. The form/table to be linked requires a key field that relates records in the two forms/tables and can be used to join them together. The keys in the main and related tables or forms do not require the same name. If the table was created in Form Designer and the data entry completed using Enter, Classic Analysis can establish a relationship automatically using the Global Record ID and Foreign Key variables created by Epi Info 7. If the relationship was created in another program that uses different keys, the key variables in both files must be identified. Relationships are represented by double colons (::). Classic Analysis can establish relationships using multiple keys. Currently, only Epi Info 7 projects can be related in Classic Analysis.
After issuing the RELATE command, the variables in the related table may be used as if they were part of the main table. Variable names are duplicated in the related tables; variable names will be suffixed with a sequence number. Frequencies, cross-tabulations, and other operations involving data in the main and related tables can be performed. The WRITE command can create a new table containing both sets of data. More than one table can be related to the main table by using a series of RELATE commands.
Relate keys can contain mathematical or string concatenation expressions. Epi Info 7 functions can be used to determine relationships. It may be easier, however, to write out a new file in which the complex key is included as a single variable. Use this single variable as a key.

PRJ File RELATE Syntax

Use READ to open the first PRJ file, and RELATE to open the second and create the relationship.

```
READ {C:\My_Project_Folder\Sample\Sample.prj}:Surveillance
RELATE RHepatitis GlobalRecordId :: GlobalRecordId
```

Try It

Epi Info 7 has a related database in the Sample.PRJ file which contains two forms: Surveillance and RHepatitis. The variable GLOBAL RECORD ID is the internal Epi Info 7 identification key located in more than one form in the project.

1. Read in the **Sample.PRJ** project. Open **Surveillance**.
2. Click **RELATE**. The RELATE dialog box opens.
3. Select **RHepatitis**. The Build Key button activates.

4. Click **Build Key**. The RELATE-BUILD KEY dialog box opens. Make sure the **Current Table** radio button is selected.
5. From the Available Variables drop-down list, select **GLOBALRECORDID**.
6. Select the **Related Table** radio button.
7. From the Available Variables drop-down list, select **GLOBALRECORDID**.
8. Click **OK**. The Related Tables field populates.
9. Click **OK**. The RELATE dialog box opens and the Key field populates with the join information.

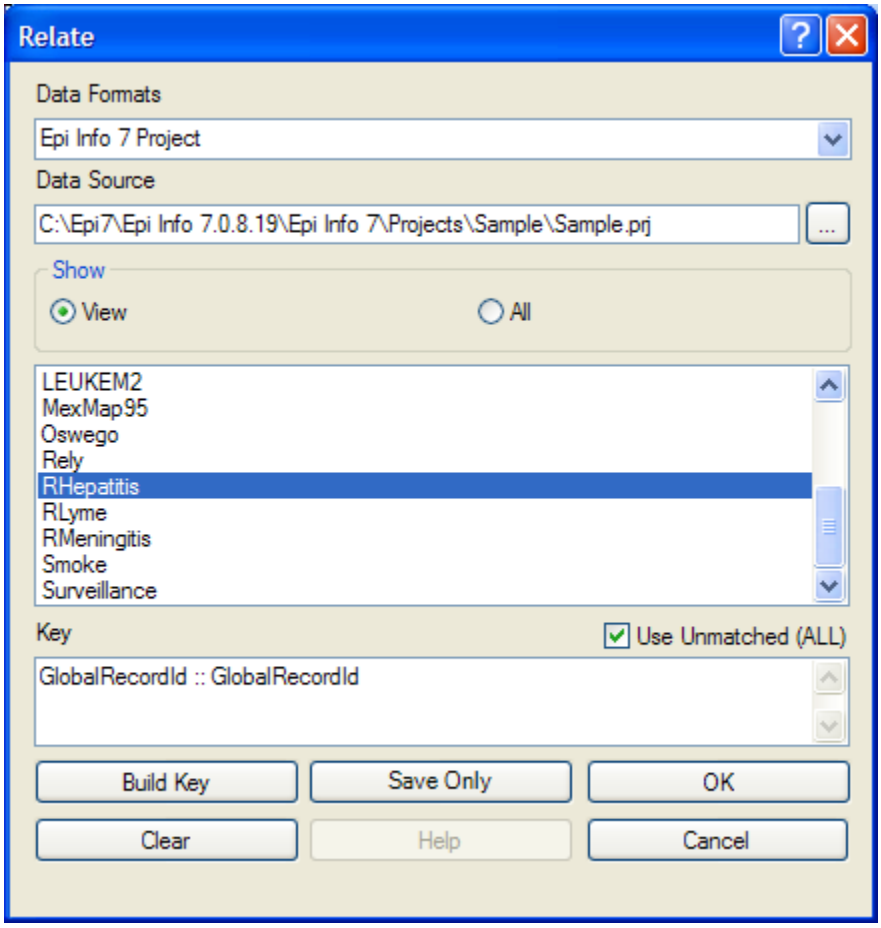


Figure 9.3: Relate Dialog Box

10. Click **OK**. The related form information appears in the Output window.
    - Data from the two tables can now be used to compute statistics.

Use the WRITE Command

Use to create a new file with selected variables.

Syntax

```
WRITE <METHOD> {<output type>} {<project>:}table {[<variable(s)>]}

WRITE <METHOD> {<output type>} {<project>:}table * EXCEPT {[<variable(s)>]}
```

To use the WRITE command, open a **project** using the READ command. Follow these steps:

1. From the Classic Analysis Command Tree, click **Write**. The WRITE dialog box opens.

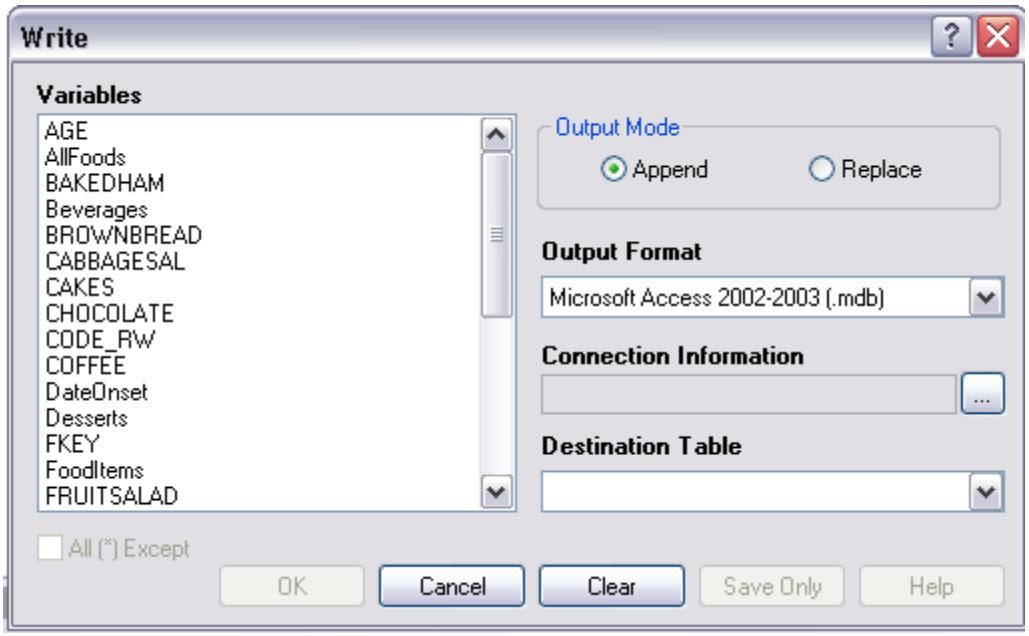


Figure 9.4: Write Dialog Box

2. Keep the default setting **All (*)** unchecked to write all the variables to a new file.

   Note: You can also choose individual variables from the file list by clicking on the variable name or select **All (*) Except** to exclude specific variables.
3. From the Output Mode section, click the **Replace** radio button.

   - **Replace** creates a new file and overwrites any existing variables and records. To overwrite or replace data in a table, use the Replace selection.
   - **Append** adds new variables and records to the end of existing data.
4. Using the drop-down list, select the desired **Output Format**.

- Available output formats are the same as ones that can be imported using the READ command (i.e., MS Access, MS Excel, SQL Server and Flat ASCII file).

5. Click the **Browse** button in the Connection Information field to specify file name and location.
6. Click **OK**.
7. Establish a **name** for the Destination Table or select a **table name** from the drop down list (if the table already exists).
8. Click **OK**.

- To see the data, use the READ command to open the newly-created project.
- The WRITE command will not create a form and cannot create a data table to work with a form. To preserve forms, use the MERGE command.
- Not all output formats support all possible input formats.

Use the MERGE Command

Use the MERGE command to join records. MERGE is only supported if the READ data source is an Epi Info 7 project. A merge requires a key, called the GLOBAL RECORD ID, which represents an internal matching variable inside both sets of data.

Syntax

```
MERGE <table specification>  <key(s)> <type>
```

1. From the Classic Analysis Command Tree, use the READ command to open **a PRJ project file**.
2. From the Classic Analysis Command Tree, click **Data > Merge**. The MERGE dialog box opens.

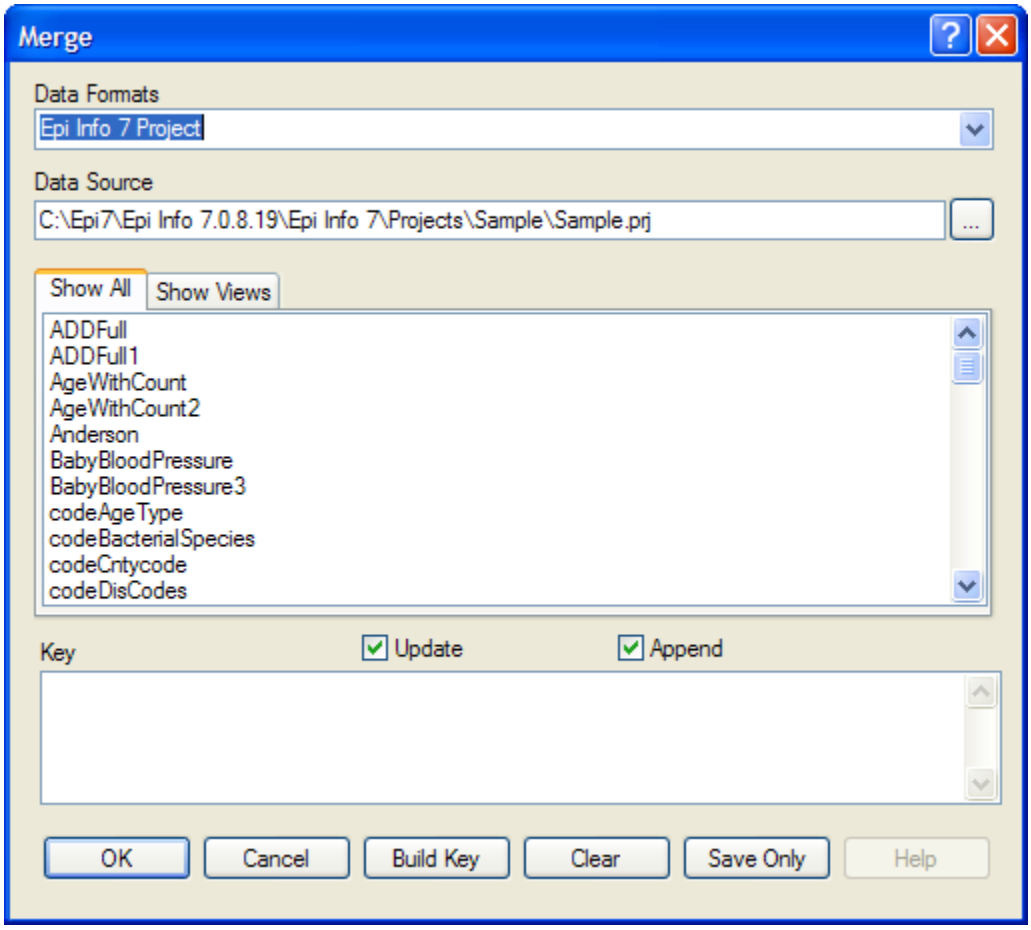


Figure 9.5: Merge Dialog Box

3. From the Data Formats drop-down list, specify the other Epi Info 7 project to be read. (Other data sources will be supported in a future release).

4. In the Data Source field, click the **Browse** button. The Merge File Name dialog box opens.

5. Select the **project file** that contains the data to be merged.

6. Click **Open**. The MERGE dialog box populates with available tables or worksheets.

7. Merge a **data table** or **worksheet**. The Build Key button activates.

8. Click **Build Key**. The RELATE - BUILD KEY dialog box opens.

    - There must be a variable in the Current Table that corresponds to a one in the Related Table (i.e., Patient ID).

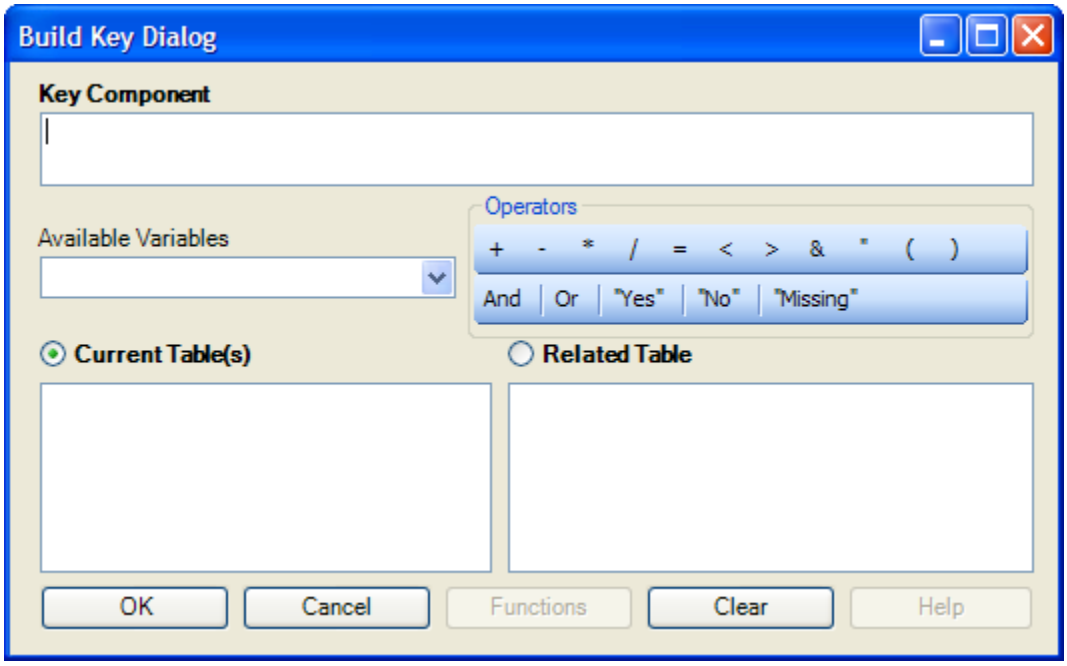


Figure 9.6: Build Key Dialog Box

9. From the Available Variables drop-down list, select the **current table merge variable.**

10. Click **OK**. The Current Table field shows the selection.

11. From the Available Variables drop-down list, select the **related table merge variable.**

12. Click **OK**. The Related Tables field shows the selection.

13. Click **OK** to accept the Build Key designations. The MERGE dialog box opens.

14. Make **Update** and **Append** selections from the corresponding checkboxes.

    - For matching records**,** Update replaces the value of any field in the READ table whose build key matches the MERGE table.
    - For unmatched records, Append creates a new record in the READ table with values only for those fields that exist in the MERGE table.
    - For most merges, select **Update** and **Append** records. If you select both options, records containing the same selected merge variables will be updated (overwritten) if there is new information. All other records will be appended (added) to the end of the data table.

15. From the Merge dialog box, click **OK**.

    - If the MERGE table is the same project as the READ table, the Record Count in the Output window updates to reflect any new records added to the table.

Use the ASSIGN Command

This command assigns an expression result or the field value to a variable. Variables are usually created with the DEFINE command and assigned a value.

Syntax

```
ASSIGN <variable> = <expression>
```

1. From the Classic Analysis Command Tree, click **Variables > Assign**. The ASSIGN dialog box opens.

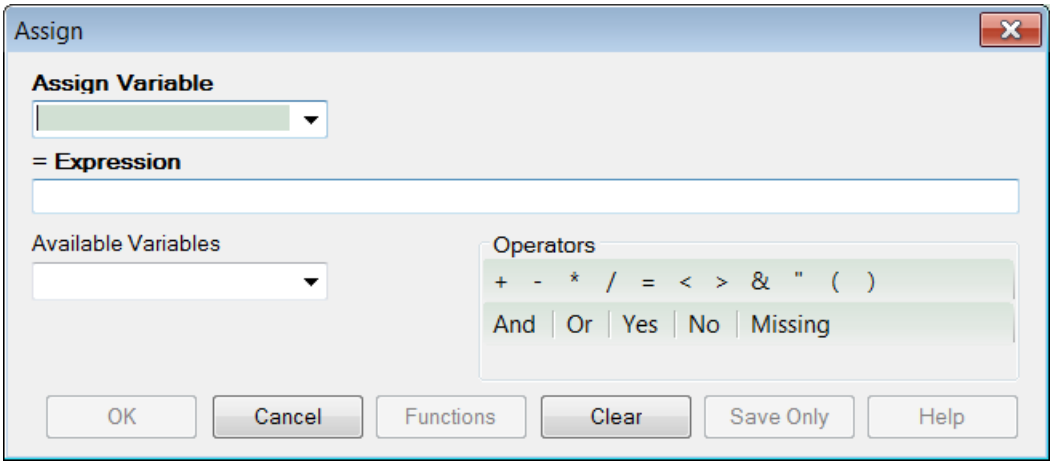


Figure 9.7: Assign Dialog Box

2. From the Assign Variable drop-down list, select the **variable** to have a value assigned.
3. In the =Expression field, create the **assignment syntax** based on needed data.
4. Click **OK**. The code appears in the Program Editor.

Try It

In the following example, the zip code field is a number. To use the zip code in a map, a new zip code variable must be defined and assigned text values.

1. From the Classic Analysis Command Tree, use the READ command to open the **Sample.PRJ project**.
2. From the Form section, click **Surveillance**.
3. Click **OK**.
4. Click **Variables > Display**.>--**Variables currently available**.
5. Click **OK**. The variables information appears.
6. Click **Variables > Define**. The DEFINE dialog box opens.

7. In the Variable Name field, create a new variable named **Zip2**.
8. Select **Text** for Variable Type
9. Click **OK**.
10. From the Classic Analysis Command Tree, click **Variables > Assign**. The ASSIGN dialog box opens.
11. From the Assign Variable drop-down list, select **Zip2**.
12. In the =Expression field, type the syntax **FORMAT(ZipCode,"00000")**.
    - In this example, the FORMAT function converts the format of the values of the variable ZipCode into text format. Text values are always surrounded by double quotes and assigned to the new Zip2 variable.

10. Click **OK**. The code appears in the Program Editor.
11. Use the DISPLAY command to view variable information.

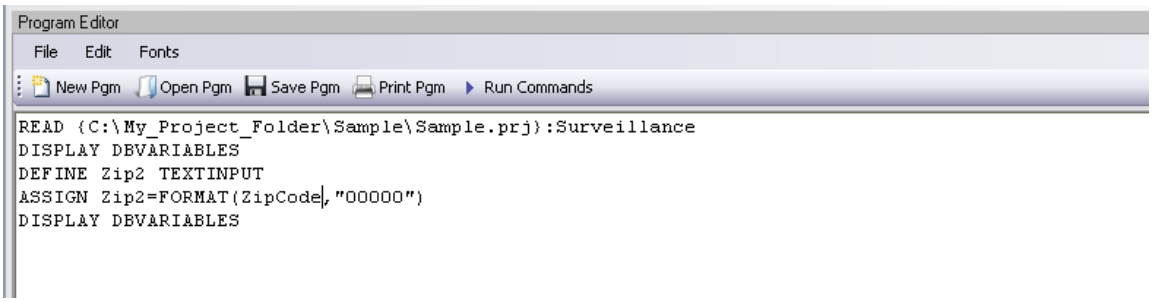


Figure 9.8: Variable Information in Program Editor

Delete File/Table

Delete Files

The file(s) specified explicitly or implicitly (via wildcards) are deleted. If no files are specified, or if any specified files cannot be deleted, a message is produced unless you select Run Silent. Wildcards are not allowed in the suffix.

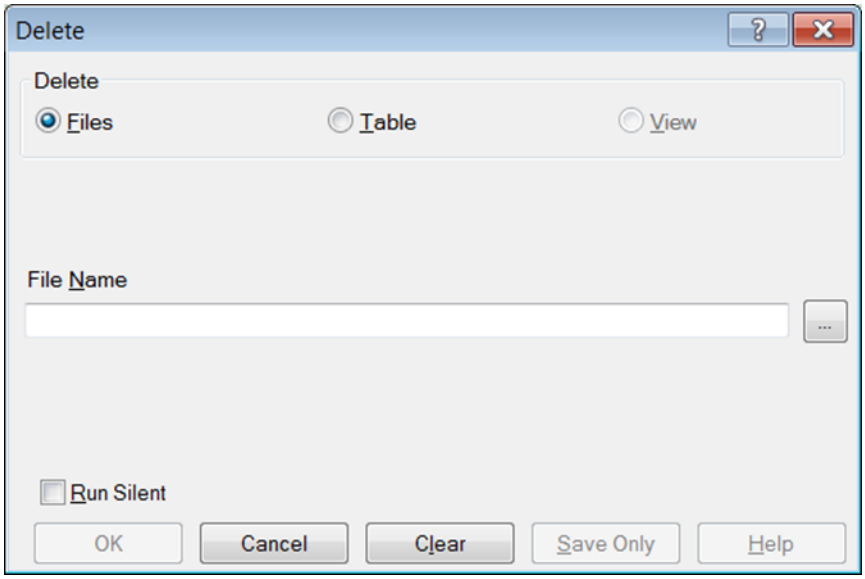


Figure 9.9: Delete Dialog Box

- **File Name** contains the name of the file to be deleted.
- **Run Silent** causes the command to run without any warning or error messages from the Program Editor.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields to allow information to be re-entered.
- **Help** opens the Help topic associated with the module being used (Currently Disabled).

Delete Table

The specified table is deleted. If the table does not exist or cannot be deleted, a message is produced unless you select Run Silent. To delete tables with spaces in their names, specify the file and the table even if the table is in the current project. The table must be enclosed in single quotes. It is possible to read a table with a space in its name and then delete it, resulting in errors during subsequent procedures.

DELETE TABLES will not delete and produce messages if any of the tables specified are data, grid, or program. Code tables are deleted only if they are not referenced by any form.

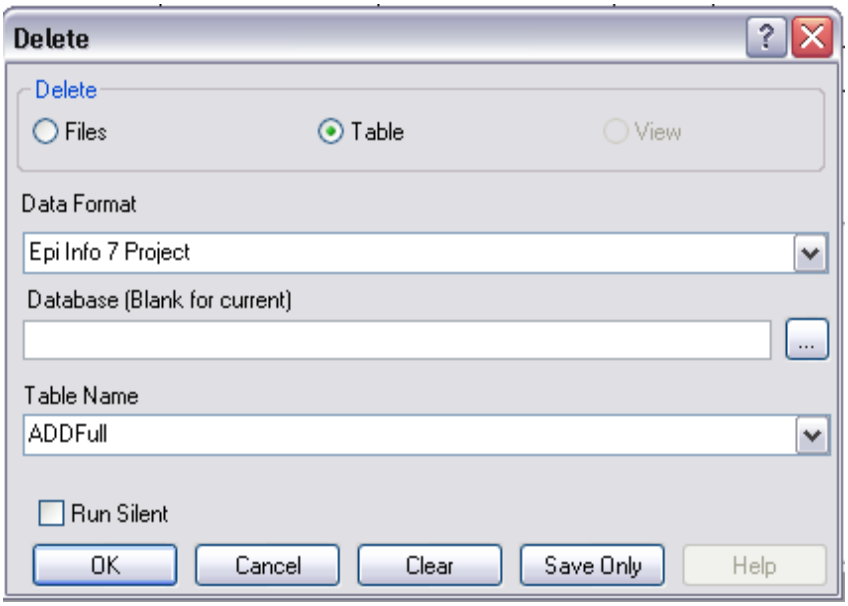


Figure 9.10: Delete Tables Dialog Box

- **Data Format** specifies the data source format of the file containing the table to be deleted.
- **Database (Blank or Current)** specifies the name and location of the dataset containing the table to be deleted.
- **Table Name** specifies the name of the table to be deleted.
- **Run Silent** causes the command to run without any warning or error messages from the Program Editor.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields to re-enter information.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Delete Records

Delete Records may not be used with related tables. For DELETE * and DELETE TABLES, space will not be reclaimed. You can run the Epi Info 3.5.3 Compact Database utility to reclaim space. All records in the current selection matching the expression are set to deleted status, or if PERMANENT is specified, physically deleted. Unless you select RUNSILENT, a confirmation message is displayed.

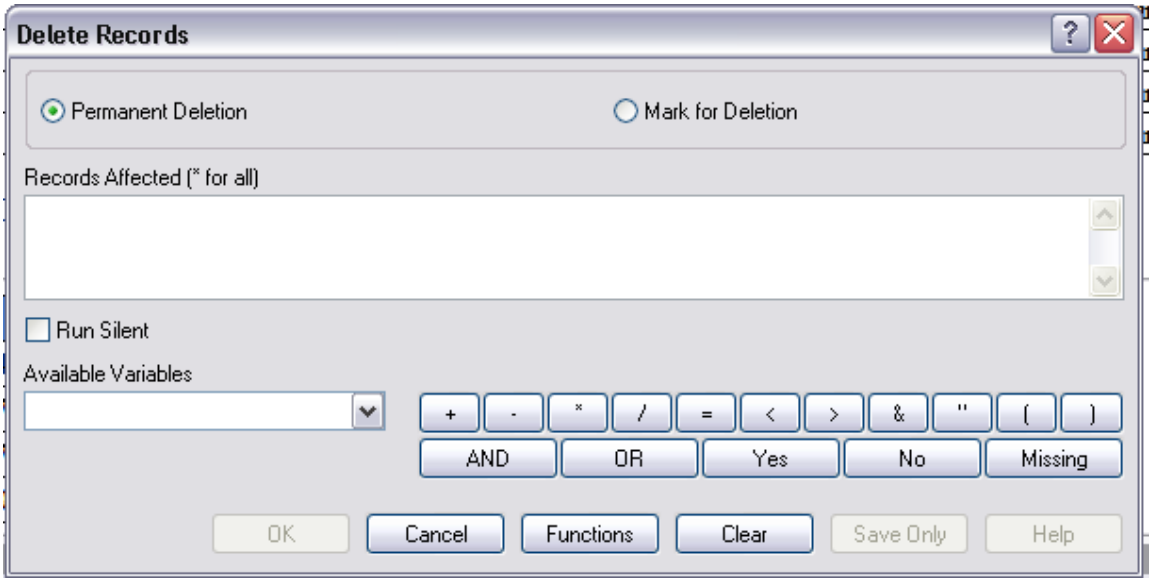


Figure 9.11: Delete Records Dialog Box

- **Permanent Deletion** causes records to be inaccessible after deletion.
- **Mark for Deletion** marks the selected records as to be deleted, and permits you to undelete. This feature can only be applied to Epi Info 7 projects.
- **The Records Affected (* for all)** field specifies which records to delete.
- **Run Silent** causes the command to run without any warning or error messages from the Program Editor.
- The **Available Variables** drop down allows you to select variables available in the current project.
- Functions and operators appear within commands and are used for common tasks (e.g., extracting a year from a date, combining two numeric values, or testing logical conditions).
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.

- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields to re-enter information.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Use the ROUTEOUT Command

You can locate output by using the ROUTEOUT command. If no directory exists, the file is placed in the current project's directory. Results accumulate until you execute a CLOSEOUT command. Output files can be placed in any folder. The ROUTEOUT command selects a path and filename. If no output file is selected, Classic Analysis uses the default value. In each folder, Classic Analysis creates a new index table that contains links to the files created.

1. From the Classic Analysis Command Tree, use the READ command to open a **PRJ project file.**

2. From the Classic Analysis Command Tree, click **Output > RouteOut**. The ROUTEOUT dialog box opens.

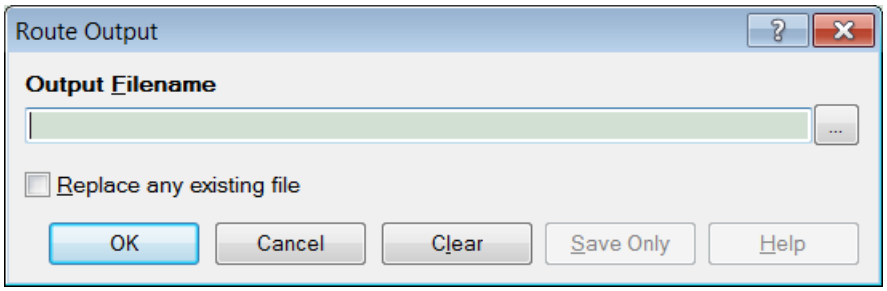


Figure 9.12: Route Output Dialog Box

3. In the Output Filename field, enter a **file name** to locate an existing file.

   - If necessary, select **Replace Existing File** to write over any files with the same name.

4. Click **OK**. Create **Output** on the existing project.

   - Notice the title bar contains the output filename specified in the ROUTEOUT command.

   - The new file is placed in the selected directory with the extension .HTM.

5. From the Output window toolbar, click **Open**. The Browse dialog box opens.

6. Locate and select the file created with ROUTEOUT. Click **Open**. The Output appears in the Output window.

   - The saved Output file can be opened by any application that can read an HTML file.

To end the ROUTEOUT command, choose one of the following options.

- From the Output folder, click **Closeout**.

- READ in a **new project**.
- Close **Epi Info 7**.

Use the CANCEL SORT Command

Syntax

CANCEL SORT Command Generator
CANCEL SORT Command Reference

1. From the Classic Analysis Command Tree, click **Select/If > Cancel Sort**. The CANCEL SORT dialog box opens.

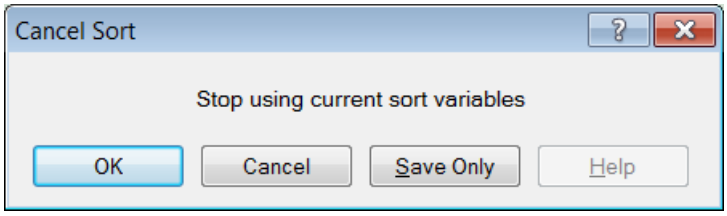


Figure 9.13: Cancel Sort Dialog Box

2. Click **OK**. The selection criteria are removed from the data, and the original record count is restored.

Undelete Records

The UNDELETE command causes all logically deleted records in the current selection matching the expression to be set to normal status. This applies only to Epi Info 7 forms and may not be used when using related tables.

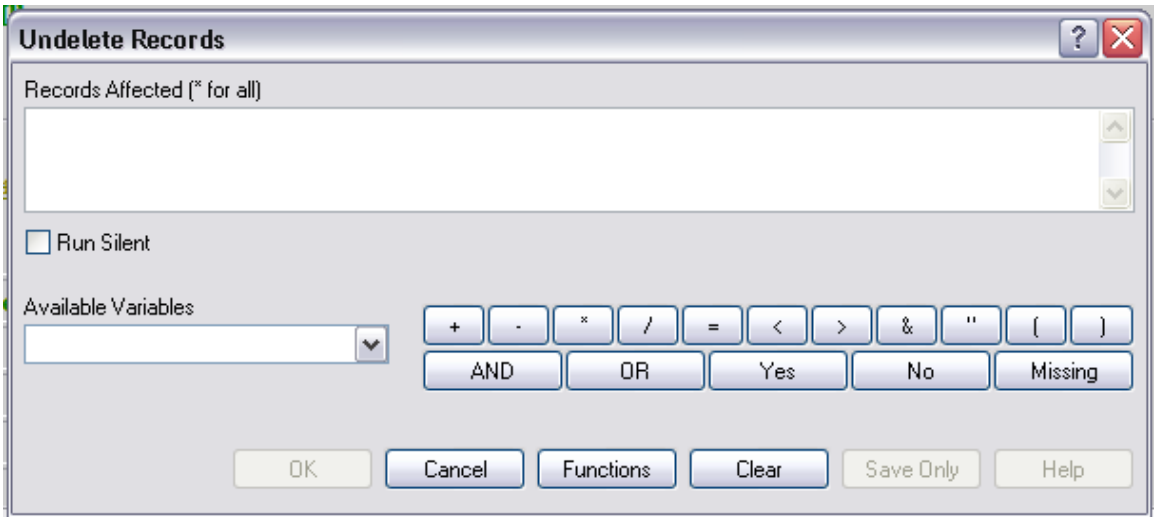


Figure 9.14: Undelete Records Dialog Box

- The **Records Affected (*for all)** field contains the expression for which records to undelete.
- The **Available Variables** field allows you to select available variables in the current project.
- Functions and operators appear within commands and are used for common tasks (e.g., extracting a year from a date, combining two numeric values, or testing logical conditions).
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Functions** opens the online help file, which explains how to use functions and operators.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields to re-enter information.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

How to Manage Variables

Use the DEFINE Command

This command creates new variables.
Syntax

```
DEFINE <variable> {<scope>} {<type indicator>} {("<prompt>")}
```

1. From the Classic Analysis Command Tree, click **Variables > Define**. The DEFINE Variable dialog box opens.

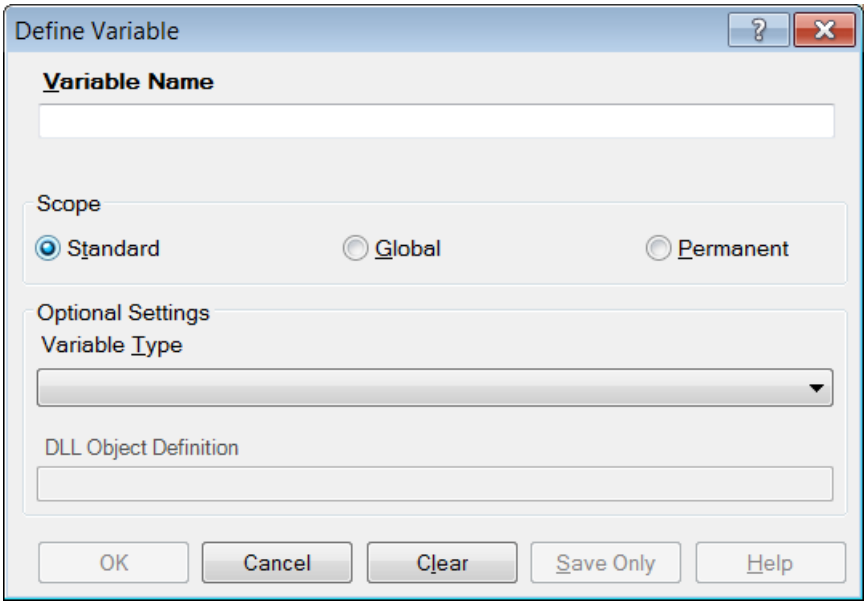


Figure 9.15: Define Variable Dialog Box

2. From the Scope section, select one of the following variables:

   - **Standard** variables remain in a current project and are one variable in that form. They can assume new values with each record during a file query (e.g., a LIST or FREQ).
   - **Global** variables persist throughout program execution. They pass values from one form to a related form, and do not change during a file query.
   - **Permanent** variables are stored in the EpiInfo.Config.xml file or system registry and retain any value assigned until changed by another assignment, or until the variable is undefined. They are shared among Epi Info 7 programs and persist even if the computer is shut down and restarted.

3. From the Variable Type drop-down list, select one of the following types: **Date**, **Numeric**, **Text**, or **Yes-No**.

4. Optional: Variable Type is required and cannot be used if <scope> is omitted. <type indicator> is the data type of the new variable and must be one of the following reserved words: NUMERIC, TEXTINPUT, YN, or DATEFORMAT. If omitted, the variable type will be inferred based on the data type of the first value assigned to the variable. However, omitting field type is not recommended.

5. Click **OK**.

Define a Group

To define a group, take the following steps:

1. From the Classic Analysis Command Tree, click **Variables > Define Group**. The DEFINE GROUP dialog box opens.

2. In the Group Variable field, type a **name**.

3. From the Variables list box, select the **variables** to be included in the group.

4. Click **OK**.

Convert a Number to Text

Numbers can be converted to text using the FORMAT function. FORMAT changes the format of one variable type to another.

Syntax

```
FORMAT(<variable>, {"<format specifcation>"})
```

Example

The Format function can be used to convert a numeric postal code or zip code field to a text type variable for mapping.

- The READ command opens a new file that contains a numeric field called zipcode. To use Epi Map with the zipcode values, change the values to text.

- The DEFINE command creates a new variable called NewZip. The variable NewZip will hold the values of zipcode in a text format. The code appears in the Program Editor as DEFINE NewZip.

- The ASSIGN command formats the variable NewZip with the values of ZipCode and the format of text. Text values are enclosed in quotes. The code appears in the Program Editor as:

```
ASSIGN NewZip=FORMAT(ZipCode,"00000")
```

- The variable NewZip can now be used as a geographic variable in Epi Map because it is a text type variable that contains numeric data.

Try It

For this example, use the Classic Analysis Program Editor to create the code.

1. Read in the **Sample.PRJ** project. Open **Oswego**. Seventy five (75) records should be listed in the Output window.

2. In the Program Editor window, place the cursor under the created Read command. Type **DEFINE MyAge TEXTINPUT**

    - Do not press the Enter key. Leave your cursor on the line of code you just created.

3. In the Program Editor window, type the following: **ASSIGN MyAge = FORMAT(Age, "00").**

4. From the Program Editor navigation menu, click **Run Commands**. The code will turn gray to indicate a syntax review.

5. From the Classic Analysis Command Tree, click **Statistics > List**. Click **OK**. The records appear.

    - Notice that the values in 'MyAge' are all two characters with a leading '0' before each single digit year. The leading zero is present because of the '00' format specification.

7. From the Classic Analysis Command Tree, click **Variables > Display**. Click **OK**. The variable types appear.

    - The variable MyAge now contains the values of Age and the format type of Text.

Use IF Statements with Permanent or Global Variables

- If the condition of an IF statement depends on global or permanent variables, the value is computed immediately and only the true or false branch, as appropriate, is followed.
- If the condition of an IF statement depends on a field variable, neither branch is followed. However, computations within the branches are saved for execution as appropriate on each record. In the second case, non-computational commands (e.g., READ, SELECT, SORT, HEADER, and ROUTEOUT) are never executed.

Handle Date Fields

Date formats are often determined by default computer settings. To access Regional Settings in a Windows environment, click **Start > Control Panel > Regional** and **Language Options**.

- Date fields in forms are entered in European, American, or ISO format as specified in Form Designer. They are stored in the database in a neutral format.
- Date fields in Excel, and Text files are read into Classic Analysis according to the date format found in Regional Settings, and stored in a neutral format. Excel files prepared with one date format often do not import properly on machines with a different date format.
- Date fields in forms are displayed in European, American, or ISO format as specified in Form Designer by the following Classic Analysis commands: LIST, LIST GRIDTABLE, TABLES, and FREQ.
- Date fields in forms are displayed according to the date format found in Regional Settings in the following Classic Analysis commands: DISPLAY, and GRAPH.
- Date fields in tables without forms are displayed according to the date format found in Regional Settings.
- Date fields written to Excel or Text files are written according to the date format in Regional Settings.
- Date literals in Form Designer, Check Code, or Classic Analysis Programs; and date-type permanent variables in the EpiInfo.Config.xml file, should be in American format unless enclosed in braces or pipe symbols. European date format (DD/MM/YYYY) literals must be enclosed in braces (i.e., {23/11/2007}). ISO date literals (YYYY/MM/DD) must be enclosed in pipe symbols (i.e.,|.,2007/11/23|.
- Dates formatted with the FORMAT function are formatted according to the date format found in Regional Settings when no format is specified as a second argument.
- Dates converted from text with TXTTODATE are converted to a neutral format according to the date format found in Regional Settings. For multi-national applications, it may be safer to use the NUMTODATE function.

If you are not sure, conduct experiments and examine the results.

How to Use Undefine

The Undefine command in Classic Analysis removes a defined variable from the system.

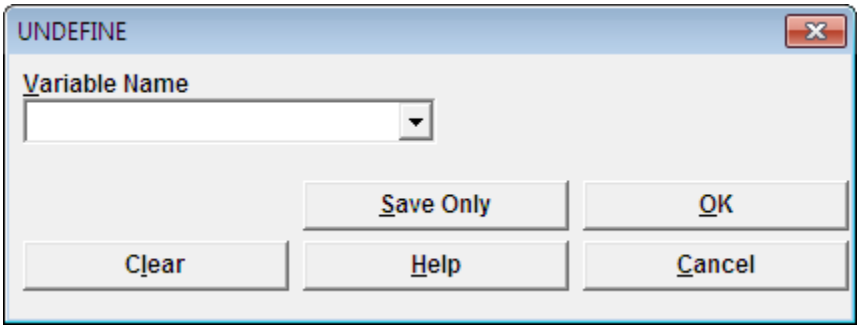


Figure 9.16: Undefine Command Box

- **Variable Name** indicates the name of the variable to be removed from the data table.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** command generation window without saving or executing a command.
- **Clear** empties fields so that information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Use the ASSIGN Command

This command assigns the result of an expression or the field value to a variable. Variables are usually created with the DEFINE command and subsequently assigned a value.

Syntax

```
ASSIGN <variable> = <expression>

<variable> = <expression>
```

1. From the Classic Analysis Command Tree, click **Variables > Assign**. The ASSIGN dialog box opens.

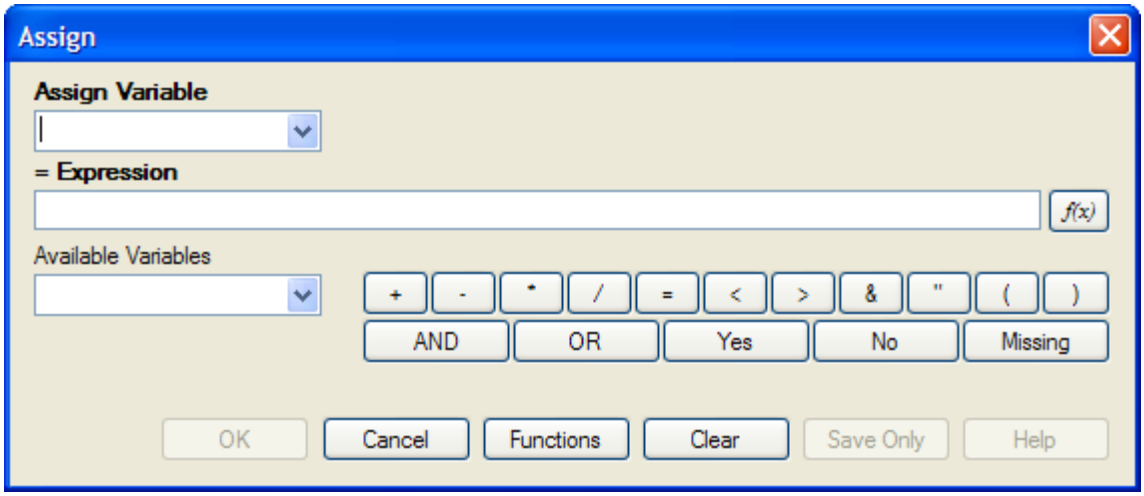


Figure 9.17: Assign Variable Dialog Box

2. From the Assign Variable drop-down list, select the **variable** to have a value assigned.
3. In the =Expression field, create the **assignment syntax** based on the needed data.
4. Click **OK**. The code appears in the Program Editor.

Try It

In the following example, the zip code field is a number. To use the zip code in a map, a new zip code variable must be defined and assigned text values.

1. From the Classic Analysis Command Tree, use the READ command to open the **Sample.PRJ project**.
2. From the form section, click **Surveillance**. Click **OK**.
3. Click **Variables > Display**. Click **OK**. The variables information appears.
4. Click **Variables > Define**. The DEFINE dialog box opens.

5. In the Variable Name field, create a new variable named **Zip2**.
6. Click **OK**.
7. From the Classic Analysis Command Tree, click **Variables > Assign**. The ASSIGN dialog box opens.
8. From the Assign Variable drop-down list, select **Zip2**.
9. In the =Expression field, type the syntax **`FORMAT(ZipCode,"00000").`**
   - In this example, the FORMAT function is used to convert the format of the values of the variable ZipCode into the text format. Text values are always surrounded by double quotes and are assigned to the new Zip2 variable.
10. Click **OK**. The code appears in the Program Editor.
11. Use the DISPLAY command to view variable information.

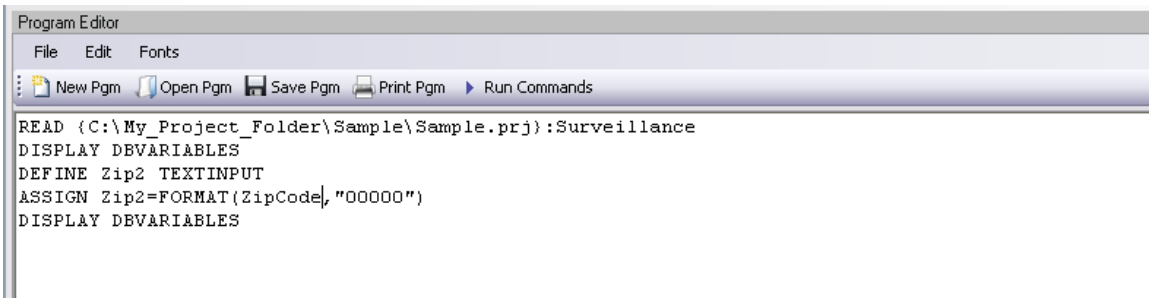


Figure 9.18: Assign Variables in Program Editor

How to Use Recode

The Recode command allows grouping of data for age and other variables, or changing code from one system to another.

- To insert a line in the table, select the **row** and press **Ctrl-Insert**.
- To delete a line in the table, select the **row** and press **Ctrl-Delete**.
- Delete any **blank rows** prior to selecting **OK** or **Save Only**.

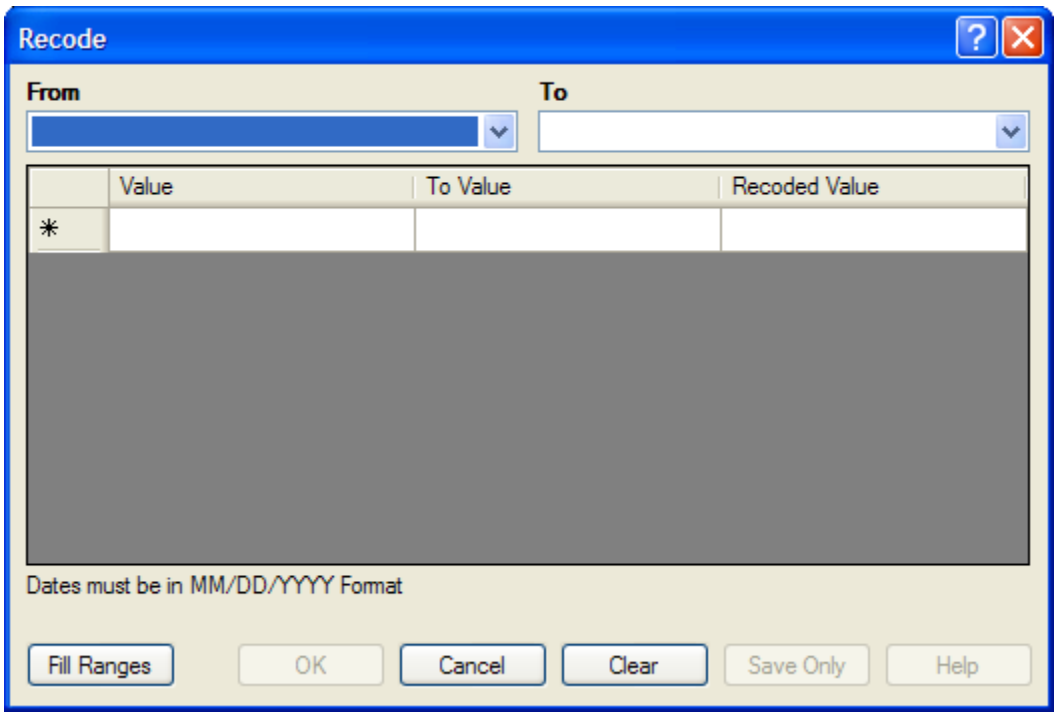


Figure 9.19: Recode Dialog Box

- The **From** drop down identifies the variable whose values are to be recoded.
- The **To** drop down identifies the variable slated to receive the recoded values.
- **Value (blank = other)** identifies the bottom of a range of values, a single value, or all remaining values slated to be recoded. Enclose text in quotation marks. The word LOVALUE may be used to indicate the smallest value in the database. HIVALUE may be used to indicate the largest. To recode a single value instead of a range, use only this column. To recode all unspecified values, leave this column and the To column blank.
- **To Value (if any)** identifies the top of a range of values that will be recoded.

- **Recoded Value** identifies the value to be assigned to the destination variable for the specified values of the source variable. Enclose text in quotation marks.
    - **Fill Ranges** allows you to redefine recoded ranges of equal distribution.
    - **OK** accepts the current settings and data, and closes the form or window.
    - **Save Only** saves the created code to the Program Editor, but does not run the code.
    - **Cancel** command generation window without saving or executing a command.
    - **Clear** empties fields so information can be re-entered.
    - **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Use the DISPLAY Command

This command displays table, form, and database information.
Syntax

```
DISPLAY <option> [<sub-option>] [OUTTABLE=<tablename>]
```

1. From the Classic Analysis Command Tree, use the READ command to open a **PRJ project file**.
    - If a .PRJ file is not opened, only language is shown for selected variables.
2. From the Classic Analysis Command Tree, click **Variables > Display**. The DISPLAY dialog box opens.

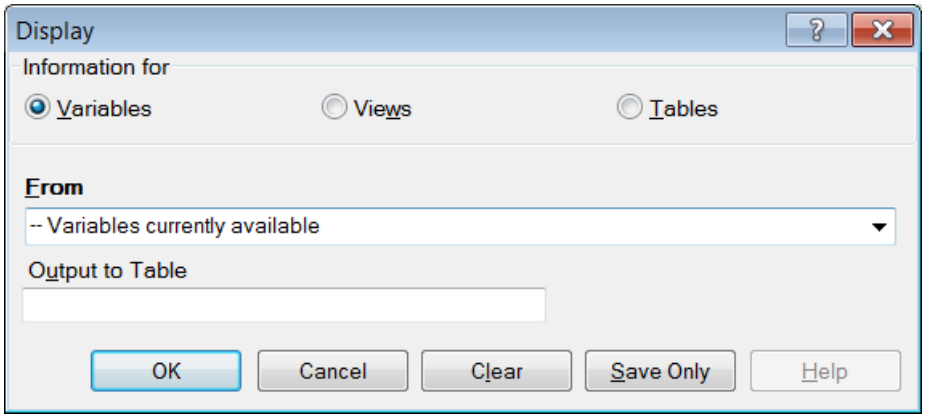


Figure 9.20: Display Dialog Box

3. From the Information for section, select one of the following form data options:
    - **Variables** displays information about all of the variables in the dataset; the defined, field, and selected variables.
    - **Forms** displays the forms in the current dataset or a selected dataset. Information about forms and other Epi Info-specific tables is shown.
    - **Tables** displays information about tables in a database. Information about tables and other Epi Info-specific or generic tables is shown.
4. If needed, type an **Output to Table** name.
5. Click **OK**. The information is displayed in the Output window.

How to Select Records

Use the SELECT Command

Use SELECT to view or analyze specific records based on selection criteria.
Syntax

```
SELECT <expression>
```

1. From the Classic Analysis Command Tree, click **Select/If > Select**. The SELECT dialog box opens.

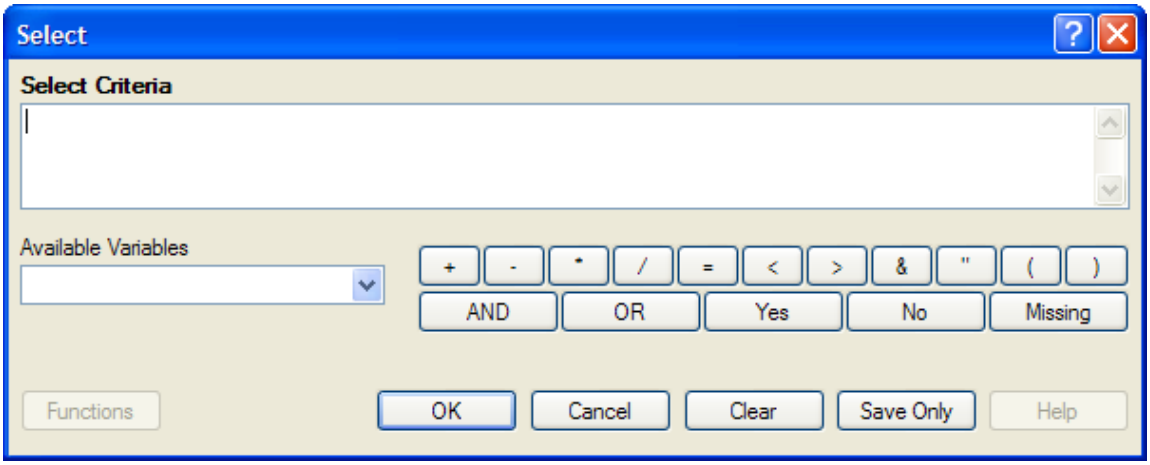


Figure 9.21: Blank Select Dialog Box

2. Use the Available Variables drop-down list to select variables from the open dataset.
3. Use the Select Criteria field and the Functions and Operators buttons to create selection code.
4. Click **OK**. The Record Count in the Output window should alter based on the number of records meeting the new criteria.
5. From the Classic Analysis Command Tree, click **Statistics > List** to view the records matching the selection criteria.
6. From the Classic Analysis Command Tree Select/If folder, click **Cancel Select** to cancel the selection criteria and return the Record Count to all records in the dataset.

Select a Date Range

1. From the Classic Analysis Command Tree, use the READ command to open a **PRJ project file**.
2. From the Classic Analysis Command Tree, click **Select/If > Select**. The SELECT dialog box opens.

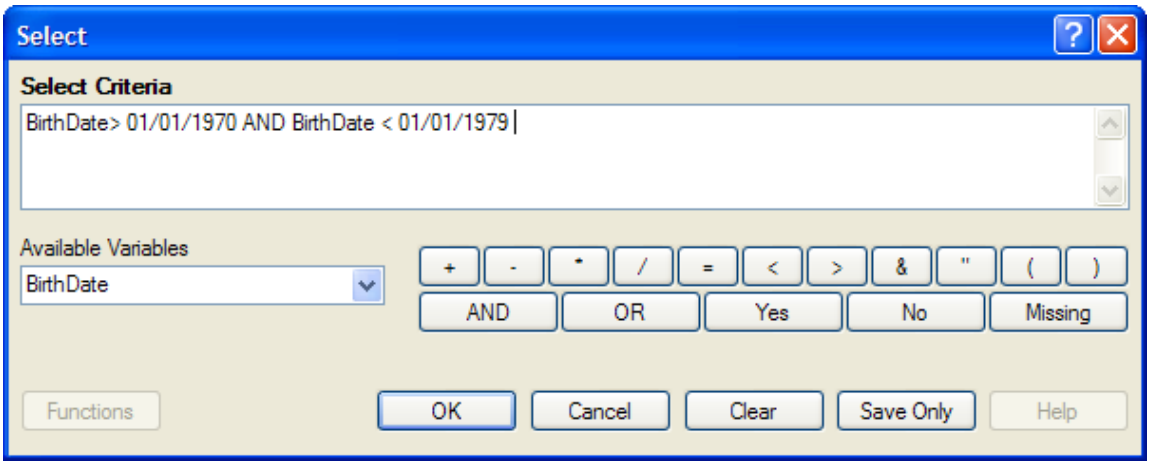


Figure 9.22: Select Dialog Box

3. From the Available Variables drop-down list, select a **date variable**.
   - In this example, DateVar is a date variable used in the data table. The DateVar has to be a date field or be converted into a date field using the NUMTODATE or TXTTONUM functions.
4. In the Select Criteria field, create the date range selection code by using the greater than >, less than <, and AND operators.

Note: Unless the date literal is enclosed in braces or pipe symbols, literal dates in PGMs are always in U.S. date format. European date format (DD/MM/YYYY) literals must be enclosed in braces (i.e., {23/11/2007}). ISO date literals (YYYY/MM/DD) must be enclosed in pipe symbols (i.e., |.,2007/11/23|).

Create the following example using the project called Sample.PRJ and the data from Surveillance.

```
READ {C:\My_Project_Folder\Sample\Sample.prj}:Surveillance
SELECT BirthDate > 01/01/1970 AND BirthDate < 01/01/1979
```

Use the CANCEL SELECT Command

SELECT
CANCEL SELECT Command Generator
CANCEL SELECT Command Reference

1. From the Classic Analysis Command Tree, click **Select/If > Cancel Select**. The CANCEL SELECT dialog box opens.

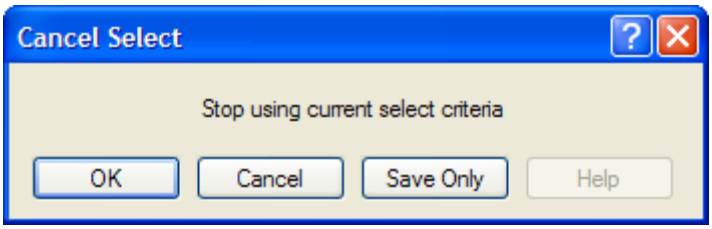


Figure 9.23: Cancel Select Dialog Box

2. Click **OK**. The selection criteria are removed from the data and the original record count is restored.

Use the IF Command

The IF command defines a condition. True causes the THEN code block to be executed; false causes the ELSE code block (if present) to be executed.

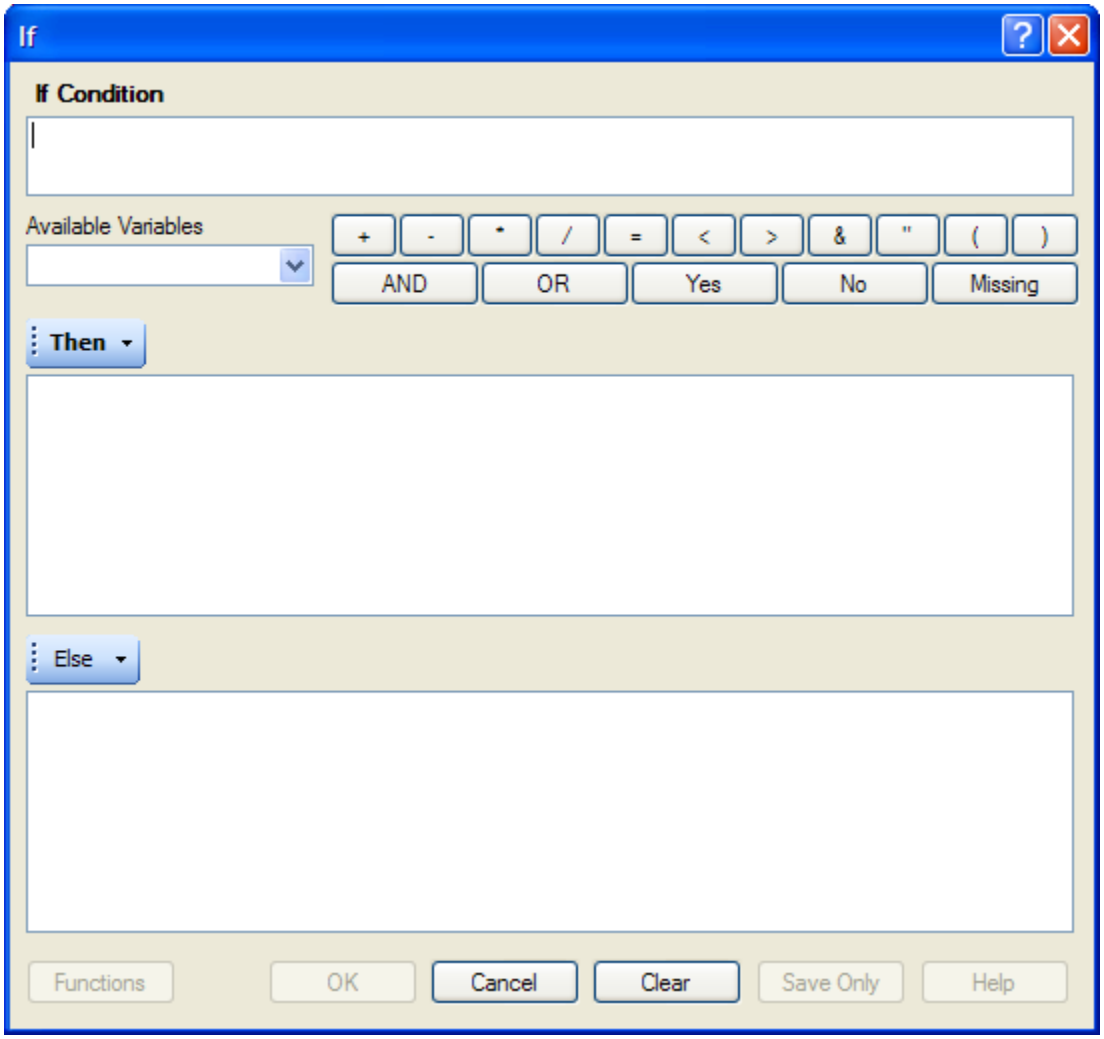


Figure 9.24: If Command Dialog Box

- The **If Condition** field specifies the condition to determine which statement that follows it is executed.
- The **Available Variables** field allows you to select variables available in the current project.
- The **Then** button is part of the If Condition statement. It starts the section of code executed when the If condition is true.

- The **Else** button is part of the If Condition and works as follows: If the condition is true, the first statement is executed. If false, the statement is bypassed, and if an else statement exists, it is executed instead.
    - Functions and operators appear within commands and are used for common tasks (e.g., extracting a year from a date, combining two numeric values, or testing logical conditions).
    - **OK** accepts the current settings and data, and subsequently closes the form or window.
    - **Save Only** saves the created code to the Program Editor, but does not run the code.
    - **Functions** opens the online Help file which explains the functions and operators. (Currently Disabled).
    - **Cancel** closes the dialog box without saving or executing a command.
    - **Clear** empties the fields so information can be re-entered.
    - **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Use the Sort Command

The Sort command allows you to specify a sequence for records to appear when using the LIST, GRAPH, and WRITE commands.

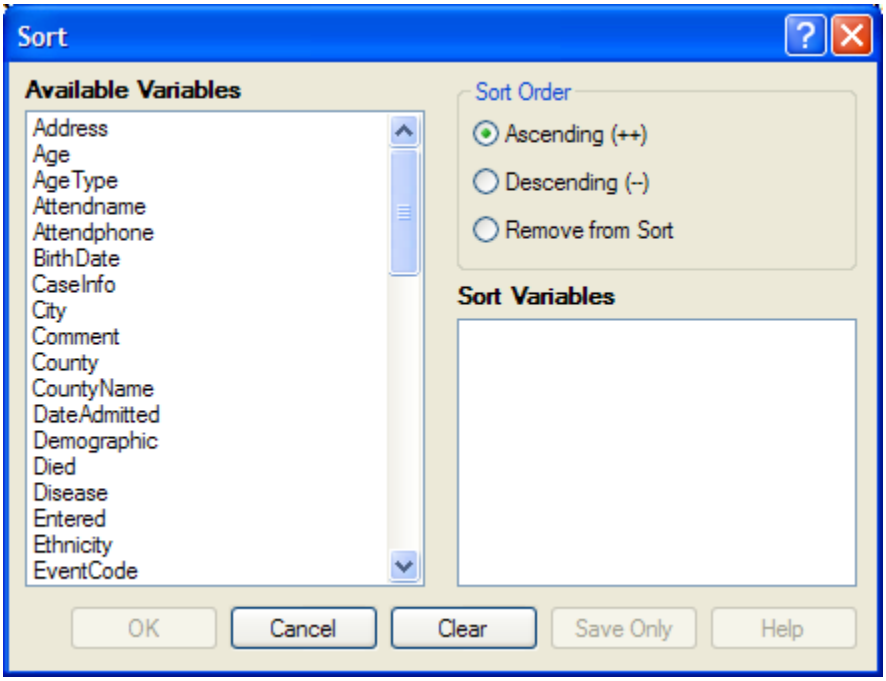


Figure 9.25: Sort Command Dialog Box

- The **Available Variables** field allows you to select variables available in the current project.
- **Sort Order** allows variables to be sorted in ascending (A to Z) or descending (Z to A) order. If the order is not specified, the sort order will be ascending. To sort one or more variables in descending order, the descending parameter (--) must be after each variable.
- Select **Remove from Sort** to remove the variables selected from the list.
- **Sort Variables** contains a list of variables selected for the sort.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so that information can be re-entered.

- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

CANCEL SORT

How to Use the CANCEL SORT command
The Cancel Sort command in Classic Analysis cancels a previous SORT command.

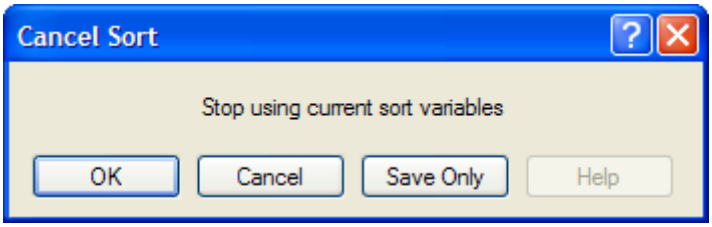


Figure 9.26: Cancel Sort

- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

How to Display Statistics and Records

Use the LIST Command

The LIST command creates a line listing of the current dataset. Select specific variables from the Variables drop-down to narrow the list or select * (the Wild Card) to display all the variables. Check the **All (*) Except** box and select from the Variables drop-down to exclude variables from the list.
To navigate LIST * GRIDTABLE results, use the Tab key to move forward one cell at a time and the Shift+Tab keys to move back one cell at a time. Navigate through the records and cells using the left, right, up, and down arrow keys.
Syntax

```
LIST {* EXCEPT} [<variable(s)>] LIST {* EXCEPT} [<variable(s)>] {GRIDTABLE}
```

1. From the Classic Analysis Command Tree, use the READ command to open a **PRJ** project file.
2. From the Classic Analysis Command Tree, click **Statistics > List**. The LIST dialog box opens.

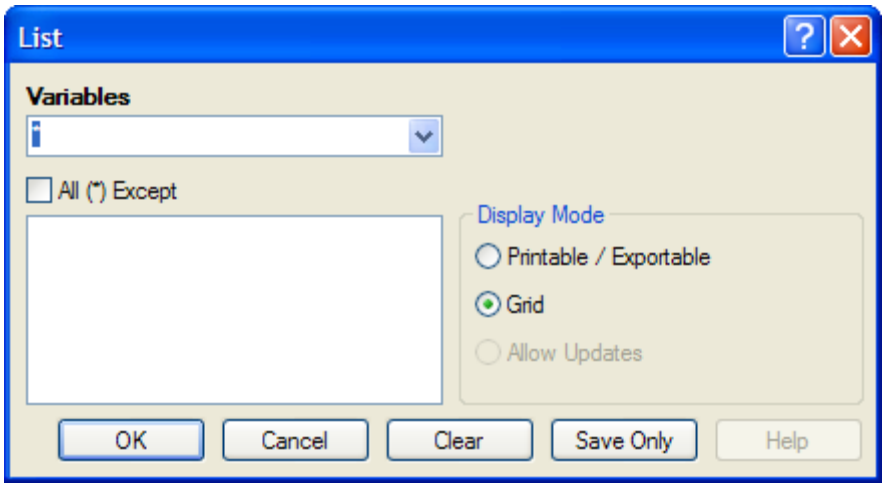


Figure 9.27: List Dialog Box

3. Click **OK** to accept the default settings. The variables in the dataset appear in a grid table inside the Output Window.
   - Grid output is not embedded. An Output file is not created.

Create a Web Format List

1. From the Classic Analysis Command Tree, click **Statistics > List**. The LIST dialog box opens.
2. From the Display Mode section, select the **Printable / Exportable** radio button.
3. Click **OK**. The List appears in the Output window in a web table format. Web display mode creates an embedded output file.

Use the FREQ Command

The FREQ command produces a frequency table that shows how many records have a value for each variable, the percentage of the total, and a cumulative percentage. The command FREQ * creates a table for each variable in the current form other than unique identifiers. This command is used to begin analyses on a new data set.

Syntax

```
FREQ [<variable(s)>]

FREQ * {EXCEPT [<variable(s)>]}
```

1. From the Classic Analysis Command Tree, use the READ command to open a **PRJ project file**.
2. From the Classic Analysis Command Tree, click **Statistics > Frequencies**. The FREQ dialog box opens.

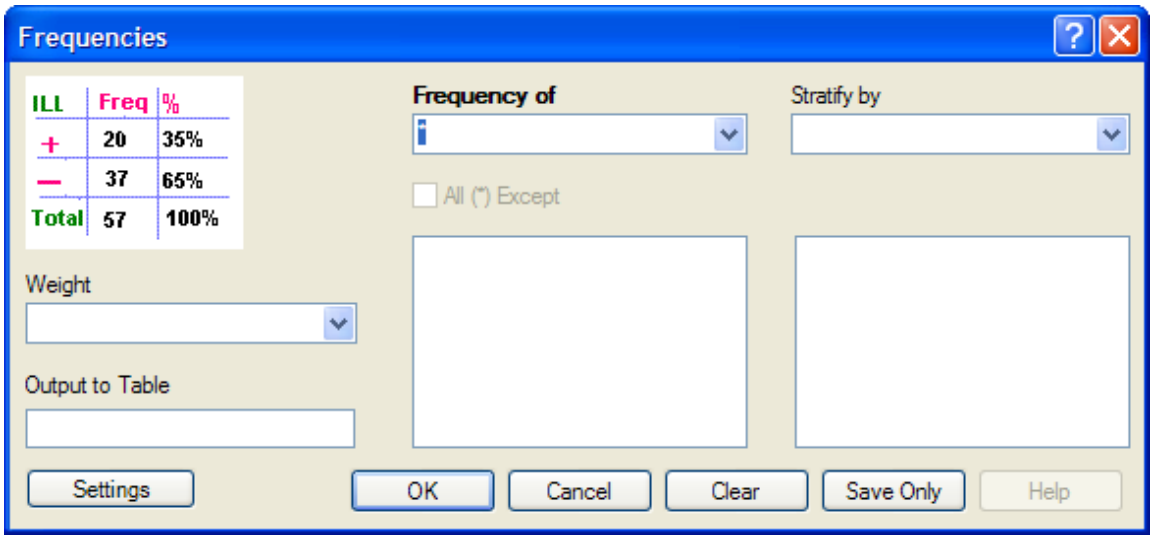


Figure 9.28: Frequency Dialog Box

3. From the Frequency of drop-down list, select a **variable** from the data table.
   - Click the **All (*) Except** checkbox to exclude variables.
   - Use the corresponding drop-down lists to select a **Weight variable** or a **Stratify by** variable from the data source.
   - Use the **Output to Table** field to specify a location for table results. The new table can be accessed using the READ command.

4. Click **OK**. Results appear in the Output window.

Try It
For this example, use the Classic Analysis Command Tree to generate the FREQ command.

1. Read in the **Sample.PRJ** project. Open **Oswego**. Seventy five (75) records should be listed in the Output window.
2. Click **Frequencies**. The FREQ dialog opens.
3. From the Frequency of drop-down list, select **ILL**.
4. Click **OK**. Results appear in the Output window.

FREQ ILL

ILL	Frequency	Percent	Cum. Percent	
Yes	46	61.33%	61.33%	
No	29	38.67%	100.00%	
Total	75	100.00%	100.00%	

95% Conf Limits

Yes	49.38%	72.36%
No	27.64%	50.62%

Figure 9.29: Frequency Output Window

- The **Frequency** column provides the count of individuals that were ill and not ill. The Percent column indicates the percentage of who were ill or not ill.
- The **95% Confidence Limits** are a range of values that indicates the likely location of the true value of a measure, meaning (in this instance) that the number of ill could be as low as 49.38%, or as high as 72.36%. Note that the Yes Ill percentage of 61.33% falls within the 95% Confidence Limits range of values based on the data.

Try It
For this example, use the Classic Analysis Program Editor to create the code.

1. Read in the **Sample.PRJ** project. Open **Oswego**. Seventy five (75) records should be listed in the Output window.
2. In the Program Editor window, place the cursor under the **Read command** created. Type **FREQ ILL**.

- Do not press the Enter key. Leave the cursor on the line of code just created.

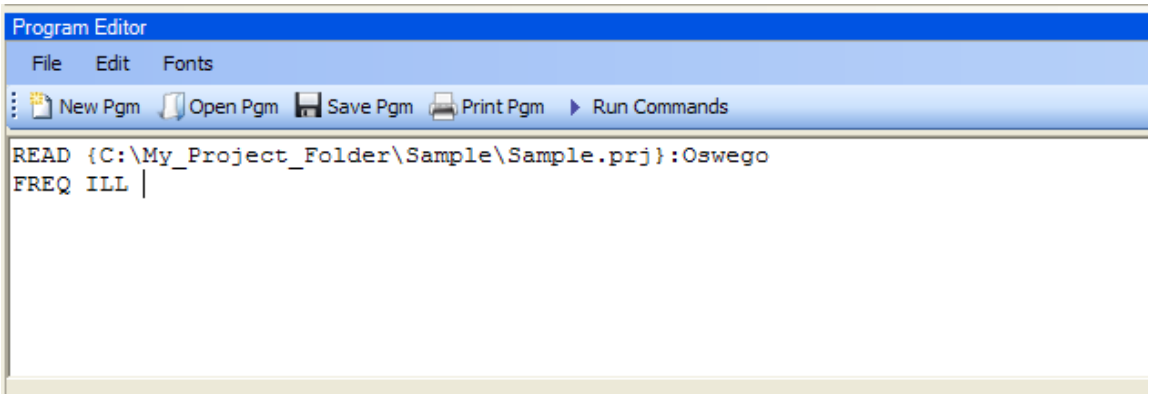


Figure 9.30: Frequency Command in Program Editor

3. Click **Run Commands** from the Program Editor navigation menu. The code turns green indicating syntax review and execution.
    - Results of the command appear in the Output window.
    - Results are the same as those created using the Command Tree dialog boxes.

Try It

A program or PGM can be saved and run repeatedly against a dataset for continuously updated statistics and analyses.

1. Read in the **Sample.PRJ** project. Open **Oswego**. Seventy five (75) records should be listed in the Output window.
2. Click **Open** from the Program Editor. The Read Program dialog box opens.
3. From the Program drop-down list, select **Statistics**.
4. Click **OK**. The PGM opens in the Program Editor.
5. Click **Run**. The Program Editor runs the code and output is placed in the Classic Analysis Output window.
    - Use the scroll bar to review all the results computed to the Output window.

Use the TABLES Command

The Tables command examines the relationship between two or more categorical values. For 2x2 tables, the Tables command produces odds and risk ratios. A 2x2 table is created when each selected variable has a yes or no answer.

Syntax

```
TABLES <exposure> <outcome> {STRATAVAR=[<variable(s)>]} {WEIGHTVAR=<variable>}
{PSUVAR=<variable>} {OUTTABLE=<table>} }
```

1. From the Classic Analysis Command Tree, use the READ command to open a **PRJ project file**.
2. From the Classic Analysis Command Tree, click **Statistics > Tables**. The TABLES dialog box opens.

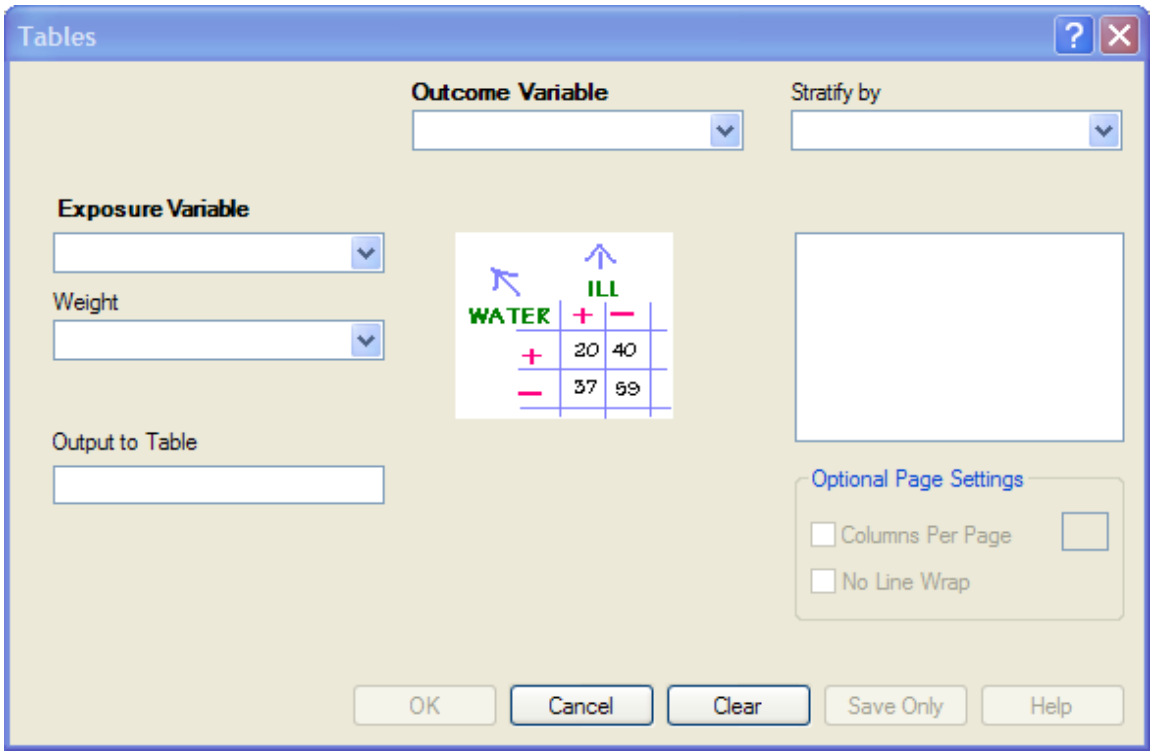


Figure 9.31: Tables Dialog Box

3. From the Exposure Variable drop-down list, select a **variable** from the data to indicate a risk factor exists.
4. From the Exposure Variable drop-down list, select a **variable** from the data to indicate the presence of an outcome. Use the:

- Stratify by drop-down list to select a **variable** to act as a grouping variable.
- Weight drop-down list to select a **variable** for weighted Classic Analysis.
- Output to Table field to specify a location for table results.
- READ command to access the READ command.

5. Click **OK**. Results appear in the Output window.
6. Scroll down to view the Single Table Classic Analysis.

Try It

Create a 2x2 table using data from the Sample.MDB project.

1. Read in the **Sample.PRJ** project. Open **Oswego**. Seventy five (75) records should be listed in the Output window.
2. Click **Tables**. The TABLES dialog box opens.
3. From the Exposure Variable drop-down list, select **Vanilla**.
4. From the Outcome Variable drop-down list, select **ILL**.
5. Click **OK**. Results appear in the Output window.

TABLES VANILLA ILL

	ILL		
VANILLA	Yes	No	Total
Yes	43	11	54
Row%	79.63%	20.37%	100.00%
Col%	93.48%	37.93%	100.00%
No	3	18	21
Row%	14.29%	85.71%	100.00%
Col%	6.52%	62.07%	28.00%
TOTAL	46	29	75
Row%	61.33%	38.67%	100.00%
Col%	100.00%	100.00%	100.00%

Figure 9.32: Tables Output Window

6. Scroll down to view the Single Table Classic Analysis.

Single Table Analysis

	Point Estimate	95% Confidence Interval Lower	Upper	
PARAMETERS: Odds-based				
Odds Ratio (cross product)	23.4545	5.8410	94.1811	(T)
Odds Ratio (MLE)	22.1490	5.9280	109.1473	(M)
		5.2153	138.3935	(F)
PARAMETERS: Risk-based				
Risk Ratio (RR)	5.5741	1.9383	16.0296	(T)
Risk Difference (RD%)	65.3439	46.9212	83.7666	(T)

(T=Taylor series; C=Cornfield; M=Mid-P; F=Fisher Exact)

STATISTICAL TESTS	Chi-square	1-tailed p	2-tailed p
Chi-square - uncorrected	27.2225		0.0000013505
Chi-square - Mantel-Haenszel	26.8596		0.0000013880
Chi-square - corrected (Yates)	24.5370		0.0000018982
Mid-p exact		0.0000001349	
Fisher exact		0.0000002597	0.0000002597

Figure 9.33: Single Tables Classic Analysis

Try It

For this example, use the Classic Analysis Program Editor to create the code.

1. Read in the **Sample.PRJ** project. Open **Oswego**
2. In the Program Editor window, place the **cursor** under the Read command created. Type **TABLES Chocolate ill**.
   - Do not press the Enter key. Leave the cursor on the line of code just created.

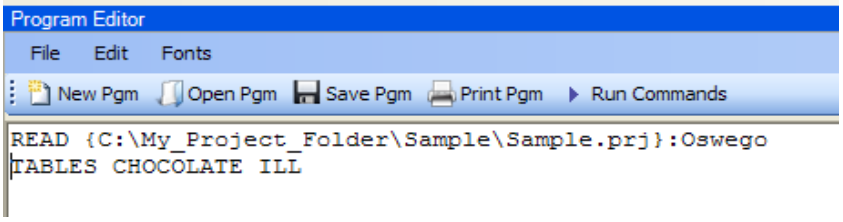


Figure 9.34: Tables Command in Program Editor

3. Click **Run Commands** from the Program Editor navigation menu. The code turns gray indicating syntax review and execution.
   - Results of the command appear in the Output window.
   - Results are the same as those created using the Command Tree dialog boxes.

Try It

A program or PGM can be saved and run repeatedly against a dataset for continuously updated statistics and analyses.

1. To see an example of a PGM, read in the **Sample.PRJ** project. Open **Oswego**.
2. From the Program Editor, click **Open Pgm**. The Read Program dialog box opens.
3. From the Program drop-down list, select **Statistics**.
4. Click **OK**. The PGM opens in the Program Editor.
5. Click **Run Commands**. The Program Editor runs the code and output is placed in the Classic Analysis Output window. Use the scroll bar to review all the results computed to the Output window.

Use the MEANS Command

The MEANS command can obtain an average for a continuous numeric variable. Since Yes equals '1' and No equals '0', the mean of a yes-no variable is the proportion of respondents answering yes. For this situation, use the FREQ command. It has two formats. If only one variable is supplied, the program produces a table similar to one produced by FREQUENCIES with descriptive statistics. If two variables are supplied, the first is numeric containing data to be analyzed. The second indicates how groups will be distinguished. The output of this format is a table similar to one produced by TABLES with descriptive statistics of the numeric variable for each group variable value.

Syntax

```
MEANS <variable 1> {<variable 2>} {STRATAVAR=<variable(s)>} {WEIGHTVAR=<variable>}
{OUTTABLE=<tablename>}
```

1. From the Classic Analysis Command Tree, use the READ command to open a **PRJ project file**.
2. From the Classic Analysis Command Tree, click **Statistics > Means**. The MEANS dialog box opens.

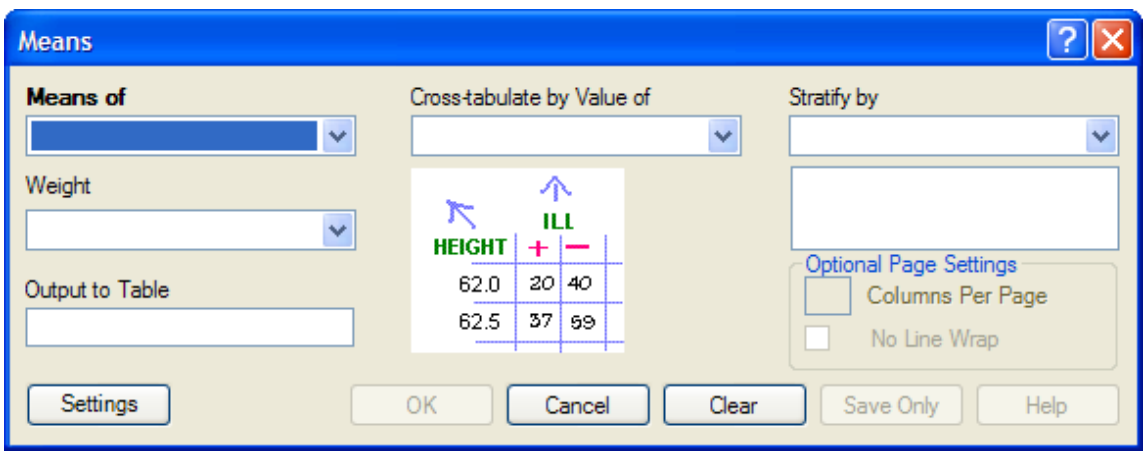


Figure 9.35: Means Dialog Box

3. From the Means Of drop-down list, select a **variable** from the data source.
   - The Cross-tabulate by value of drop-down list contains a variable to help determine if the means of a group are equal.
   - The Stratify by drop-down list contains a variable to act as a grouping variable.
   - The Weight drop-down list contains a variable for weighted Classic Analysis.

- The Output to Table field specifies a location for table results. The new table can be accessed using the READ command.

4. Click **OK**. The Output window populates with the results. Scroll down to view the Mean and Standard Deviation results.

Try It

Use the MEANS command to find the average for a continuous variable.

1. Read in the **Sample.PRJ** project. Open **Oswego**. Seventy five (75) records should be listed in the Output window.
2. Click **MEANS**. The MEANS dialog box opens.
3. From the Means of drop-down list, select **Age**.
4. Click **OK**. Results appear in the Output window.

MEANS AGE

AGE	Frequency	Percent	Cum. Percent
3	1	1.33%	1.33%
7	2	2.67%	4.00%
8	2	2.67%	6.67%
9	1	1.33%	8.00%
10	1	1.33%	9.33%
11	4	5.33%	14.67%
12	1	1.33%	16.00%
13	2	2.67%	18.67%
14	1	1.33%	20.00%
15	3	4.00%	24.00%
16	1	1.33%	25.33%
17	4	5.33%	30.67%
18	1	1.33%	32.00%
20	2	2.67%	34.67%

Figure 9.36: Means Output Window

5. To view the Descriptive Statistics, scroll down the Age listing.
   - Notice the mean age of individuals in the dataset is 36.8.

Obs	Total	Mean	Variance	Std Dev
75.0000	2761.0000	36.8133	460.1809	21.4518

Minimum	25%	Median	75%	Maximum	Mode
3.0000	16.0000	36.5000	58.0000	77.0000	11.0000

Figure 9.37: Output Data

Try It
For this example, use a numeric variable and a Yes/No variable to determine the MEANS for a group.

1. Read in the **Sample.PRJ** project.
2. Click **MEANS**. The MEANS dialog box opens.
3. From the Means of drop-down list, select **Age**.
4. From the Cross-Tabulate by Value of drop-down list, select **ILL**.
5. Click **OK**. Results appear in the Output window.
6. To view the Descriptive Statistics, scroll down the Age listing.
    - Notice the Mean Age of those answering yes to ill is 39.2. Those answering no to ill is 32.9.

Descriptive Statistics for Each Value of Crosstab Variable					
	Obs	Total	Mean	Variance	Std Dev
No	29.0000	955.0000	32.9310	423.7094	20.5842
Yes	46.0000	1806.0000	39.2609	477.2638	21.8464

	Minimum	25%	Median	75%	Maximum	Mode
No	7.0000	15.5000	35.5000	52.0000	69.0000	11.0000
Yes	3.0000	17.5000	37.0000	59.5000	77.0000	15.0000

Figure 9.38: Descriptive Statistics Output Window

Other available statistics include:
- ANOVA, a Parametric Test for Inequality of Population Means.
- Bartlett's Test for Inequality of Population Variances.
- Mann-Whitney/Wilcoxon Two-Sample Test (Kruskal-Wallis test for two groups).

Try It
For this example, use the Classic Analysis Program Editor to create the code.
1. Read in the **Sample.PRJ** project. Open **Oswego**.
2. In the Program Editor window, place the cursor under the Read command created. Type **MEANS Age**.
    - Do **not** press the Enter key. Leave the cursor on the line of code just created.

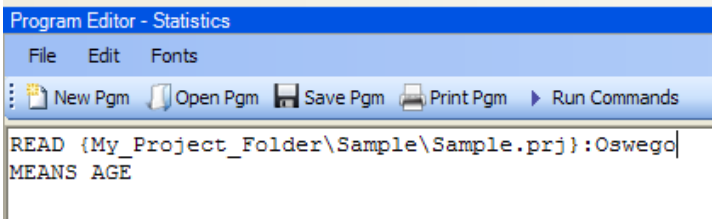


Figure 9.39: Means Age in Program Editor

3. Click **Run Commands** from the Program Editor navigation menu. The code turns gray to indicate execution of syntax.
    - Results of the command appear in the Output window.
    - Results are the same as those created using the Command Tree dialog boxes.

Use the SUMMARIZE Command

The SUMMARIZE command aggregates data producing an output table showing the descriptive statistics.

Syntax

```
SUMMARIZE varname::aggregate(variable) [varname::aggregate(variable) ...] TO tablename
STRATAVAR=variable list {WEIGHTVAR=variable}
```

1. From the Classic Analysis Command Tree, use the READ command to open a **PRJ project file**.
2. From the Classic Analysis Command Tree, click **Statistics > Summarize**. The Summarize dialog box opens.

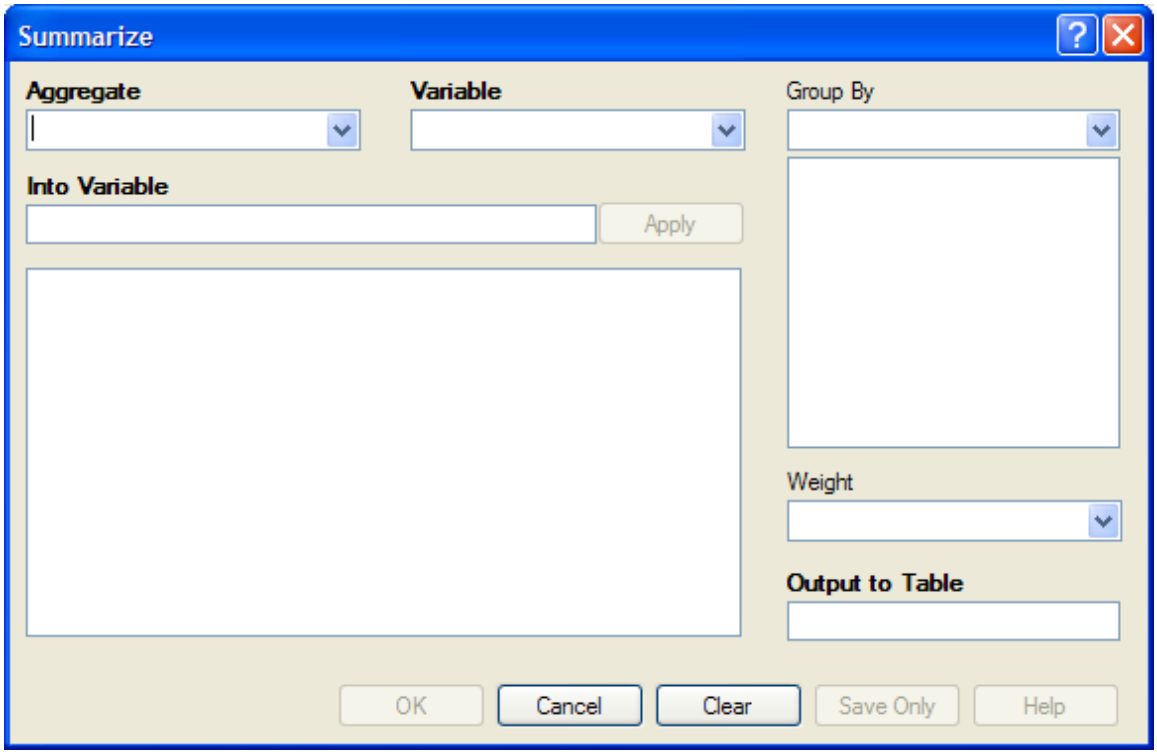


Figure 9.40: Summarize Dialog Box

3. From the Aggregate drop-down list, select from the available functions to specify how records will be combined. The options are:
   - **Average -** used only on numeric fields.
   - **Count**

- **First -** based on sorts order.
- **Last -** based on sorts order.
- **Maximum**
- **Minimum**
- **StdDev -** Standard Deviation.
- **StdDev(Pop)**
- **Sum -** only used on numeric fields.
- **Var-** Variance - only used on numeric fields.
- **Var(Pop)**

4. From the Variable drop-down list, select from the **variables** in the dataset.
5. In the Into Variable field, type a **name** to title the summarized variable.
6. Click **Apply**. The code appears in the open field.
   - If grouping is necessary, use the Group By drop-down list to select a **variable** to group the aggregated data.
   - If a weight variable exists, use the Weight drop-down list to select a **variable** to weight the aggregated data.
7. In the Output to Table field, type a **name** for the new data table.
8. Click **OK**. The following message appears in the Output window: Output table created: location listed.
9. Click **Read/Import**. The READ dialog box opens.
10. Click the **All** radio button. All the tables in the dataset appear in a list.
11. Locate and select the **summary table** just created.
12. Click **OK**. The Output window populates with the Record Count for the summary table.
13. Use the LIST command to view the summary table.

Try It

1. Read in the **Sample.PRJ** project. Open **ADDFull**. Three hundred and fifty nine (359) records should be listed in the Output window.
2. Click **Statistics > Summarize**. The SUMMARIZE dialog box opens.
3. From the Aggregate drop-down list, select **Average**.

4. From the Variable drop-down list, select the variable **GPA**.
5. In the Into Variable field, type **AverageGPA**.
6. Click **Apply**. The code appears in the open field.
7. From the Group By drop-down list, select the variable **GENDER**.
8. In the Output to Table field, type **DATATABLE1**.
9. Click **OK**. The following message appears in the Output window:

```
SUMMARIZE AverageGPA :: Avg(GPA) TO DATATABLE1 STRATAVAR = GENDER
```

10. Click **Read/Import**. The READ dialog box opens.
11. Click the **Tables** checkbox. All the tables in your dataset appear in a list.
12. Select the summary table **DATATABLE1**.
13. Click **OK**. The Output window populates with a record count of two for the summary table.
14. Use the LIST command to view the summary table.

Use the GRAPH Command

The following steps create a basic graph containing one main variable. The GRAPH command opens the Epi Graph application and creates a variety of graph types based on the types of data available in the project.

Syntax

```
GRAPH [<Variable(s)>] GRAPHTYPE ="<GraphType>" [<OptionName>=OptionValue]

GRAPH [<Variable(s)>] * <Crosstab> GRAPHTYPE ="<GraphType>" [<OptionName>=OptionValue]

GRAPH [<Variable(s)>] GRAPHTYPE="<GraphType>" XTITLE="<string>" YTITLE="<string>"
```

1. From the Classic Analysis Command Tree, use the READ command to open an Epi Info 7 **.PRJ file**.
2. From the Classic Analysis Command Tree, click **Statistics > Graph**. The GRAPH dialog box opens.
   - The default graph selection is Bar.

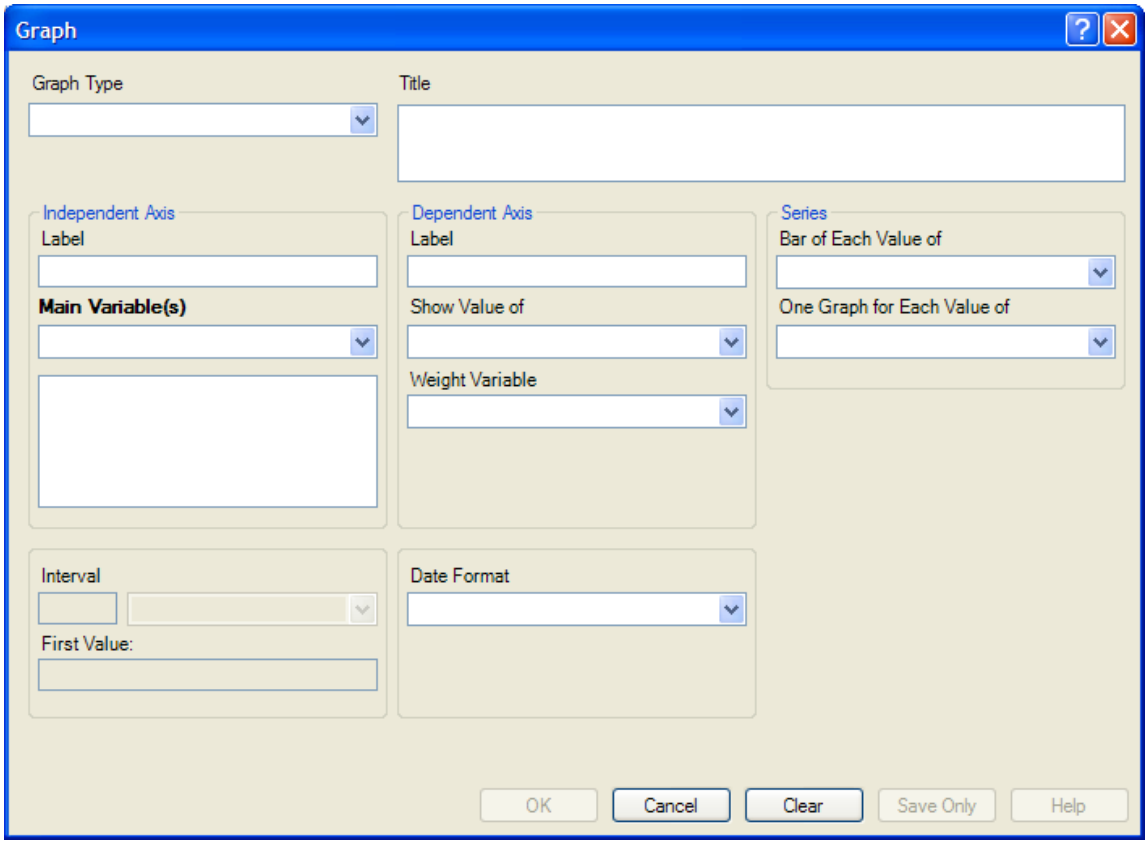


Figure 9.41: Graph Dialog Box

3. From the Graph Type drop-down list, select a type of **graph** from the list.
4. From the Main Variable(s) drop-down list, select the **X-Axis value** to graph. Use the:
    - Show Value Of drop-down list to select an **aggregate** when multiple records need to be displayed as a group.
    - Weight Variable drop-down list to select a **variable** for weighted Classic Analysis.
    - Bar for Each Value Of drop-down list to select a **variable** to generate multiple series from a single data source, each with a different color or style.
    - One Graph for Each Value of drop-down list to select a **variable** to generate a multiple series from a single data source, each displayed on a separate set of axes.
    - X-Axis Label box to type a **label** to appear in the graph.
5. Click **OK**. Epi Graph opens.
6. View the graph. Graphs can be saved by clicking on the floppy disk icon located on the top right hand side of the graph.
7. The graph is now embedded in the Output window as part of your output file.

Try It
Create a bar graph using data from the Sample.MDB project.

1. Read in the **Sample.PRJ** project. Open **Oswego**. Seventy five (75) records should be listed in the Output window.
2. Click **Statistics > Graph**. The GRAPH dialog box opens.
3. From the Graph Type drop-down list, select **Column**.
4. From the Main Variable drop-down list, select the variable **Age** to use for the X-Axis.
5. Click **OK**. Epi Graph opens.

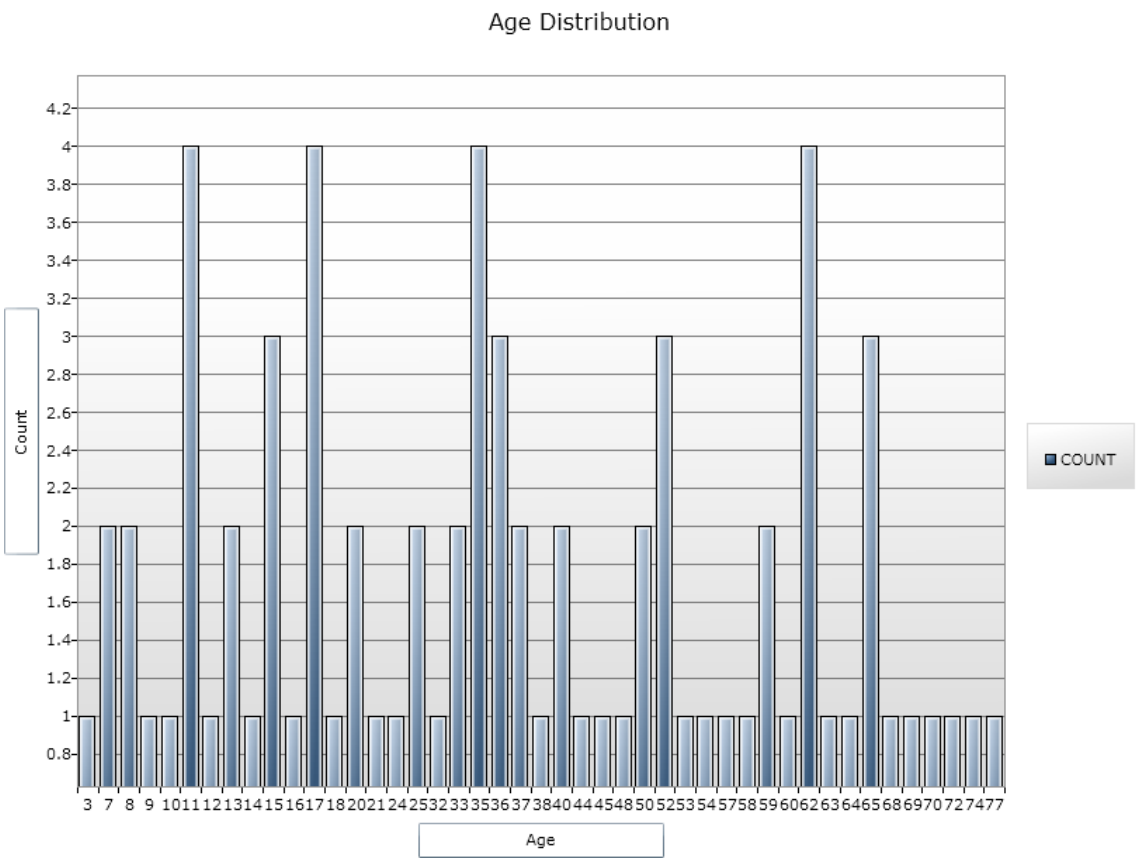


Figure 9.42: Bar Graph

6. The bar graph is now embedded in the Output window and part of the output file.
    - Notice the Graph code that appears in the Program Editor.

How to Manage Output

Header

The Header command in Classic Analysis sets up specific headings as part of the output in Analysis.

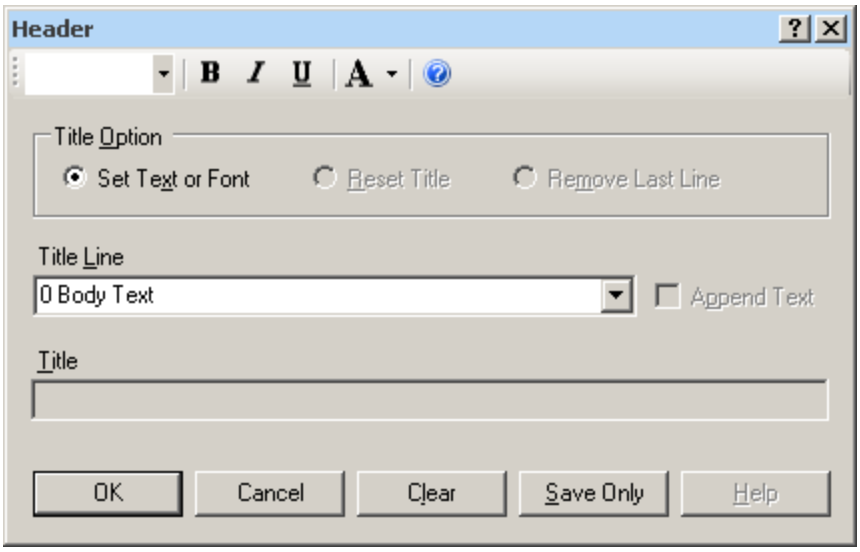


Figure 9.43: Header Command Window

- **Title Option** permits title lines to be set or changed. If you select Reset Title, the selected title level will be reset to default titles. If you select Remove Last Line, the last line of a title level will be removed.
- **Title Line** indicates which level of titles or body text is affected by the command and settings.
- **Append Text** adds a new line of text to an existing header. If not selected, the current title is replaced.
- **Title** indicates the text in the header title.
- **Bold** B displays header fonts in bold style text.
- **Italic** I italicizes header fonts.
- **Underline** U underlines header fonts.
- **Font Size** A sets the text font size.
- **Font Color** A Blue allows you to apply a color to the header text.

- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

TypeOut

The TypeOut command in Classic Analysis inserts text, either a string or the contents of a file, into the output. Typical uses may include comments or boilerplates.

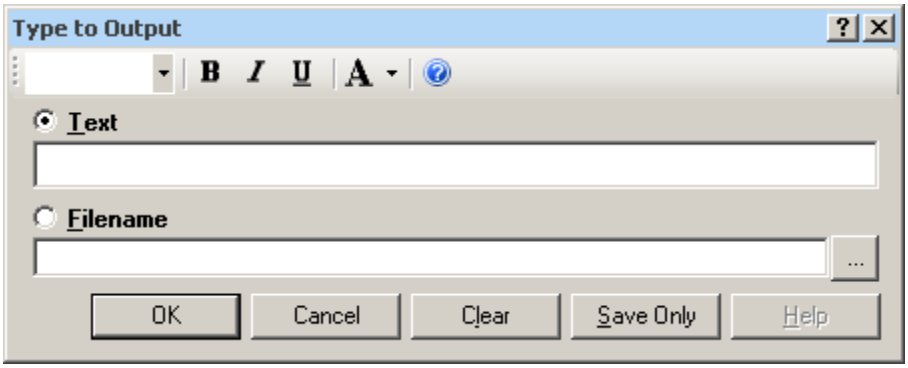


Figure 9.44: Type to Output, or TypeOut, Window

Selecting "Filename" opens the Windows File Explorer to allow you to select a file to be included in your output.

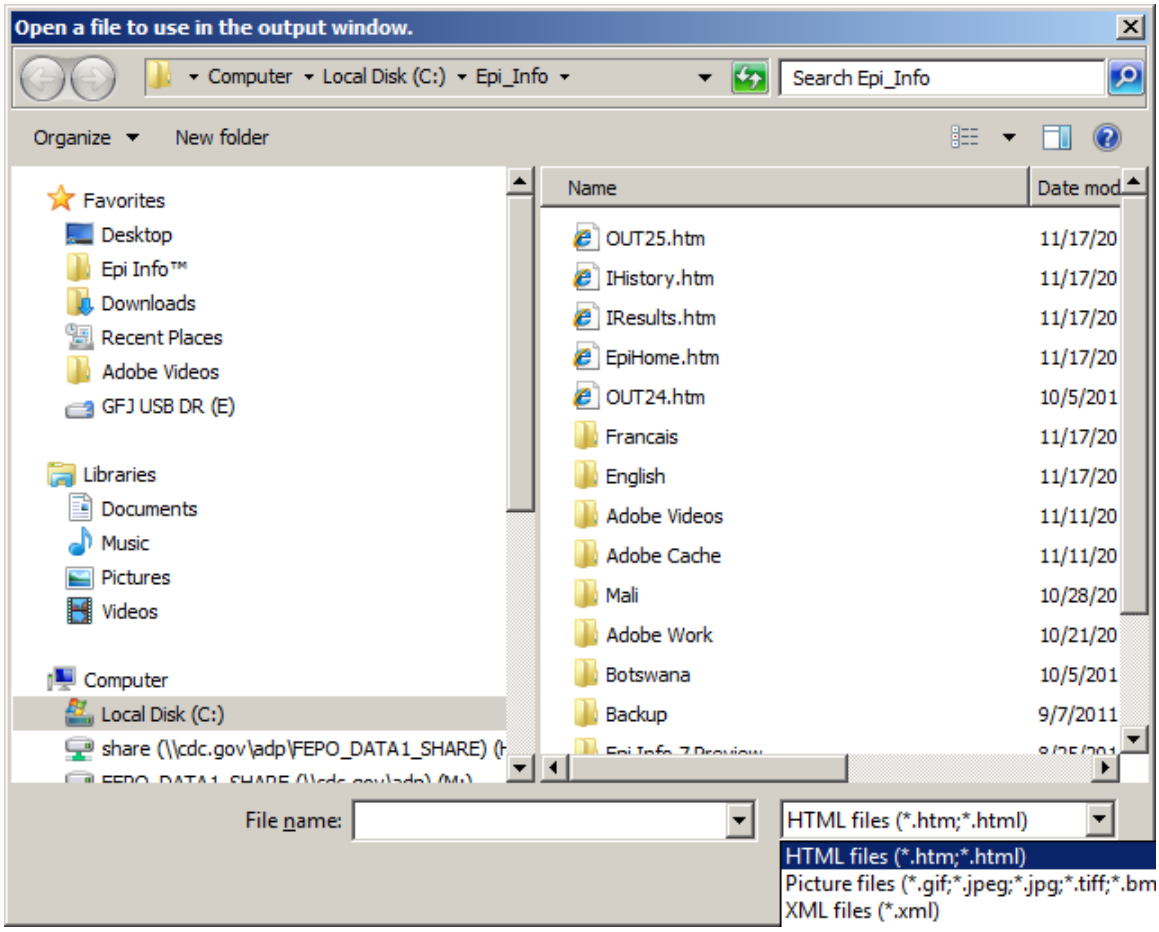


Figure 9.45: Open File Window

- The **Text or Filename** field contains the text or file name to be inserted in the output file.
- **Bold** displays type fonts in bold style text.
- **Italic** italicizes type fonts.
- **Underline** underlines type fonts.
- **Font Size** sets the type text font size.
- **Font Color** allows you to apply a color to the type text.
- Type a **filename** or click the **ellipse** to locate a file (e.g., HTM, JPEG, XML**)**.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Use the ROUTEOUT Command

The ROUTEOUT command enables you to locate output. If no directory exists, the file is placed in the current project's directory. Results accumulate until a CLOSEOUT command is executed. Output files can be placed in any folder. The ROUTEOUT command selects a path and filename. If no output file is selected, Analysis uses the default value. In each folder, Analysis creates a new index table that contains links to the files created.

1. From the Analysis Command Tree, use the READ command to open a **PRJ project file**.

2. From the Analysis Command Tree, click **Output > RouteOut**. The ROUTEOUT dialog box opens.

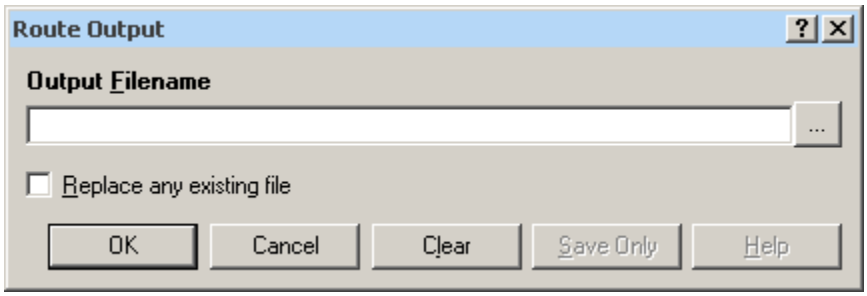


Figure 9.46: Route Output Window

3. In the Output Filename field, enter a **file name** to locate an existing file.

   - If necessary, select **Replace Existing File** to write over any files with the same name.

4. Click **OK**. Create **Output** on the existing project.

   - Notice the title bar contains the output filename specified in the ROUTEOUT command.

   - The new file is placed in the selected directory with the extension .HTM.

5. From the Output window toolbar, click **Open**. The Browse dialog box opens.

6. Locate and select the **file** created with ROUTEOUT. Click **Open**. The Output appears in the Output window.

   - The saved Output file can be opened by any application that can read an HTML file.

To end the ROUTEOUT command, choose one of the following options.

- From the Output folder, click **Closeout**.
- READ in a **new project**.
- Close **Epi Info**.

Closeout

The Closeout command closes the current output file.

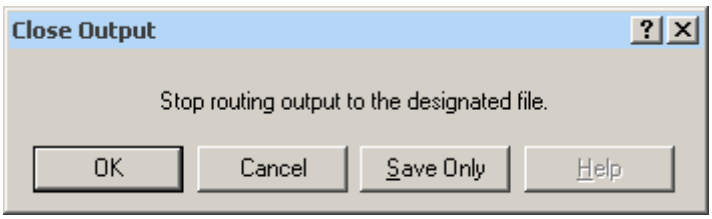


Figure 9.47: Close Output Window

- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Printout

Command Reference

The Printout command sends the current output file, or another file specified by the user, to the default printer. This differs from the print button on the output window because a command is generated that causes printing whenever it is run.

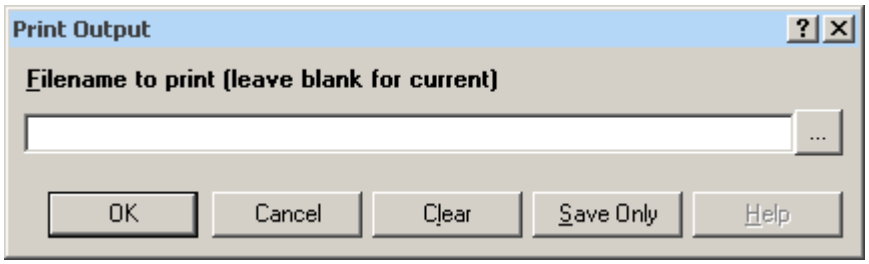


Figure 9.48: Print Output Window

- If **Filename** is selected, the file will print. If blank, it prints the current output.
- The ellipses (…) to the right of the filename opens a Windows dialog box and allows you to search the computer for the program or command to execute.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Storing Output

The Storing Output command defines how output files are stored in the current project directory with a name composed of a prefix and a sequence number.

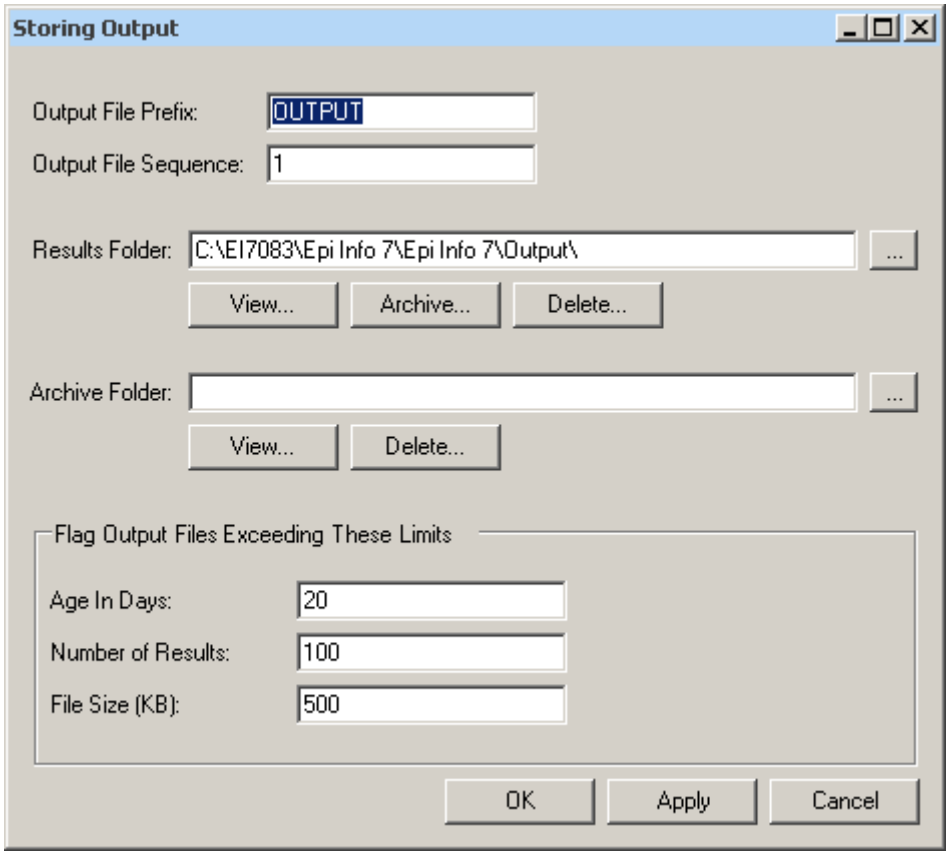


Figure 9.49: Storing Output Command Window

- **Output File Prefix** contains the first part of the filename of the output files. It is used with the Output File Sequence to store output files in the directory.
- **Output File Sequence** contains the second part of the filename for output files. This number is automatically incremented for each file. Along with the Output File Prefix, it stores output files in the directory.
- **Results Folder** locates the Results index file in the project directory.
- The **Archive Folder** allows you to locate the Archive file in the directory.
- **Archive** opens the Archive Results window, and allows specific output files to be selected and archived to the current directory.
- **Delete** opens the Delete Results window and allows you to delete specific files from the directory.

- **Age in Days** marks output files older than the indicated number of days in the form window if Flags are active. If output file limits are placed on number of days, it appears in the Flag Files operation.
- **Number of Results** marks output files in excess of the number indicated in the form window if Flags are active.
- **File Size** marks output files in excess of the size indicated in the form window if Flags are active. Enter output file size limits in this field.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Apply** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** exits the dialog box without saving or executing a command.

How to Use User-Defined Commands

Define Command (CMD)

The User Define Command allows you to create a block of code that can be invoked by name, like a subroutine. This is useful for sequences of commands that will be called more than once.

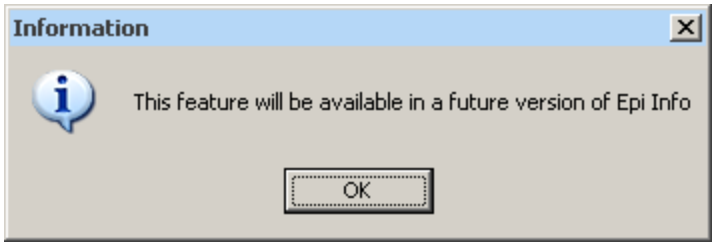


Figure 9.50: Future Version Notice

- This command will be available in a future version of Epi Info 7.

Run Saved Program

Command Reference

The **Run Saved Program** command in Classic Analysis transfers control to the second program returning to the first automatically, beginning with the line following the RUN statement. A program being run from another program is similar to an INCLUDE or subroutine in other systems.

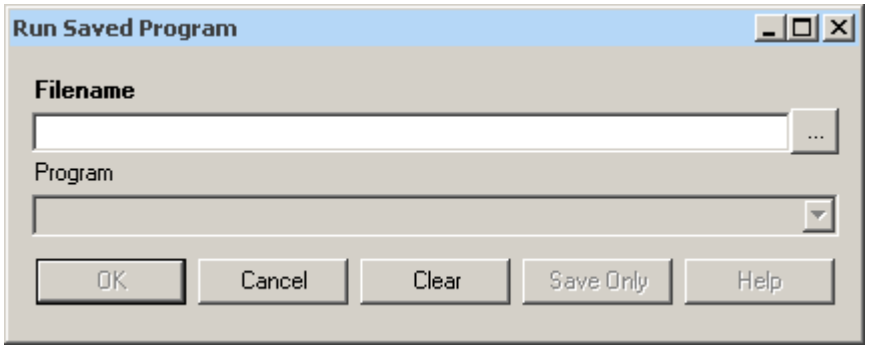


Figure 9.51: Run Saved Program

- **Filename** indicates the database or text file containing the program. If the path or filename contains a space, it must be enclosed in double quotes. If the program to be run is in the current project, the path does not need to be supplied.
- **Program** indicates the program name if it's in the database.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Save and Open a Program (PGM)

Commands entered into Classic Analysis generate lines of code in the Program Editor and can be stored in the current data source (.mdb file or SQL server). Programs can be saved internally within the project or externally as a text file with a .pgm7 file extension. Saved programs that can be run as data are updated. Programs can be saved as text files, and shared without sharing data tables or projects.

Example of Code Created in the Program Editor

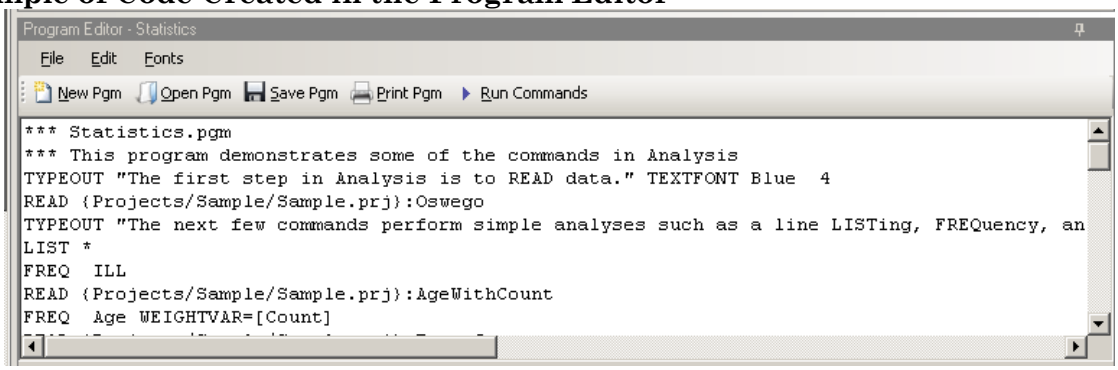


Figure 9.52: Example Code in Program Editor

Save Code as a PGM

1. From the Program Editor Navigation bar, click the **Save Pgm** icon. The Save Program dialog box opens.

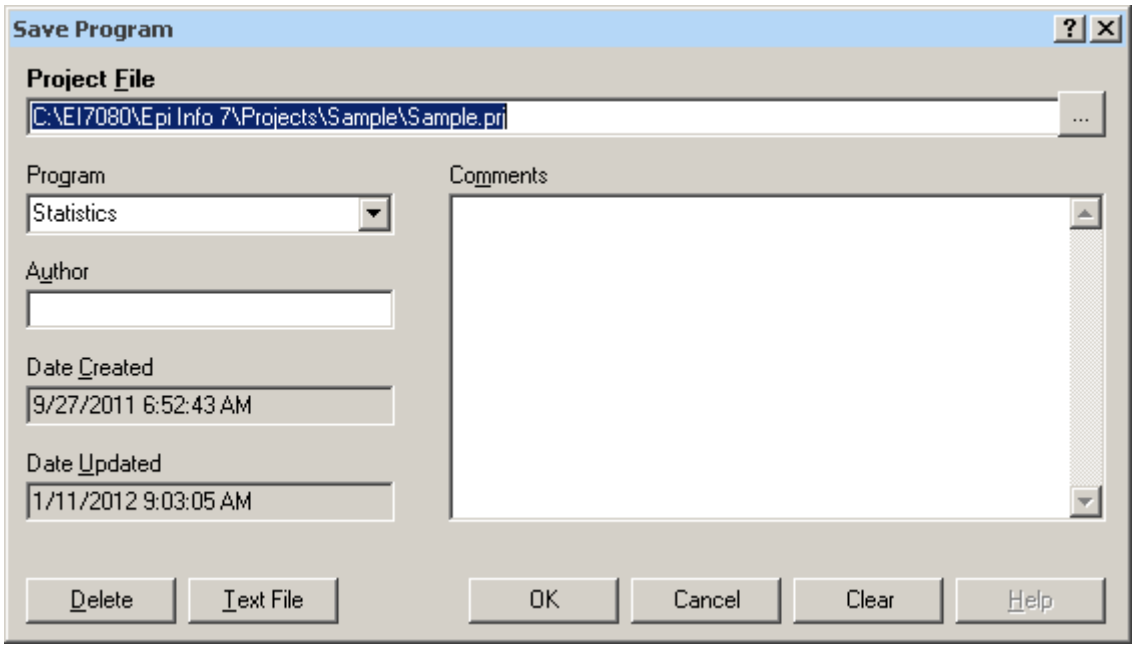


Figure 9.53: Save Program Dialog Box

2. In the Program field, type a **name** for the program.
3. In the Author field, type a **name** or **initials**.
4. In the Comments field, type a **description** of the program.
5. Click **OK**.
   - When code is saved, it is written to a special table in the current .mdb, called Programs.
   - A saved program can be executed with the RUNPGM command, or opened in the Program Editor.
   - From the Save Program dialog box, click **Text File** to save code to a text file.

Open and Run a Saved PGM

1. From the Program Editor, select **File > Open Pgm** or click the **Open Pgm** icon. The Read Program dialog box opens.

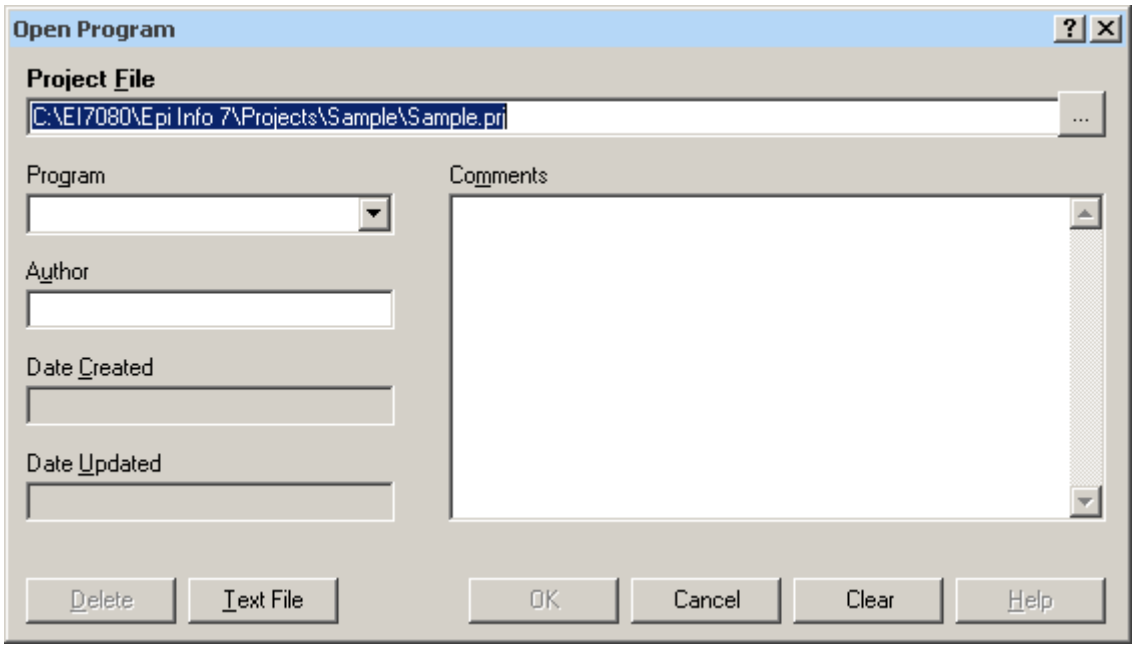


Figure 9.54: Read Program Dialog Box

2. From the Program drop-down list, select a saved **Pgm**.
   - Information about the program automatically populates the open fields in the dialog box.
   - To open a program that was saved externally, click **Text File** to view all available .pgm files.

3. Click **OK**. The program code opens in the Program Editor.
    - Code can now be run, edited, or saved.
4. Click **Run Commands**. The Program Editor runs all the code listed and displays the results in the Output window.
    - To run individual commands, use the cursor to highlight the commands you want to run. Click **Run Commands**.

Execute File

Command Reference

This command executes a Windows program in Classic Analysis. You can either explicitly name the command (e.g., WinWord.exe) or one designated within the Windows registry as appropriate for a document with the named file extension (C:\Temp\MyDocument.doc). This provides a mechanism for bringing up whatever word processor or browser is the default on a computer without first knowing its name.

If the pathname is a long filename, it must be surrounded in single quotes. If the command takes parameters, surround the command and the parameters with a single set of double quotes. Do not use single quotes.

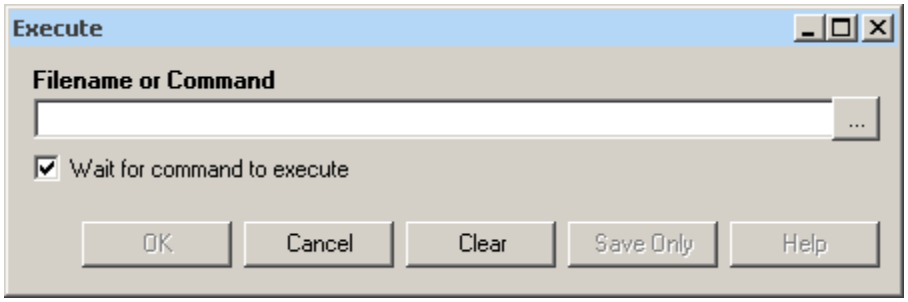


Figure 9.55: Execute File Dialog Box

- **Filename** is the file name, program name, or command to execute. Enter a path and program name for .EXE and .COM files along with any desired command line arguments.
- **Wait for command to execute** indicates whether the program should run before or after the command is executed.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Syntax

```
EXECUTE <filename>
EXECUTE <program-name>
EXECUTE "<fprogram-name><command-line parameters>"
EXECUTE NOWAITFOREXIT <filename>
EXECUTE NOWAITFOREXIT '<filename>'
EXECUTE NOWAITFOREXIT "<filename>"
EXECUTE WAITFOREXIT <filename>
```

```
EXECUTE WAITFOREXIT '<filename>'
EXECUTE WAITFOREXIT "<filename>"
```

- The <program name> represents the path and program name for .exe (filename for registered Windows programs) and .com (any Internet address) files, along with any desired command line arguments. If a parameter is added, the whole string should be enclosed within double quotes.

- If EXECUTE is run modally, permanent variables are written before the command is executed, and reloaded after it is executed.

Comments

If the given name is not a program, but a file with an extension (the three characters after the ".") registered by Windows for displaying the document, the correct program to display the file will be activated (i.e., WRITEUP.DOC might cause Microsoft Word to run and load the file on one computer). Usually .TXT will run NOTEPAD.EXE or WORDPAD.EXE, and image files will appear either in a browser or graphics program. An .HTM file will bring up the default browser.

Example

```
EXECUTE "C:\Epi_Info_7\Enter.exe sample.prj:oswego"
EXECUTE "C:\ Epi_Info_7\OUT120.htm'
EXECUTE "C:\windows\notepad.exe c:\Test1.txt"
```

How to Create User Interaction

DIALOG

How to Use the DIALOG Command

The DIALOG command provides user interaction from within a program. Dialogs can display information, ask for and receive input, and offer lists to make choices.

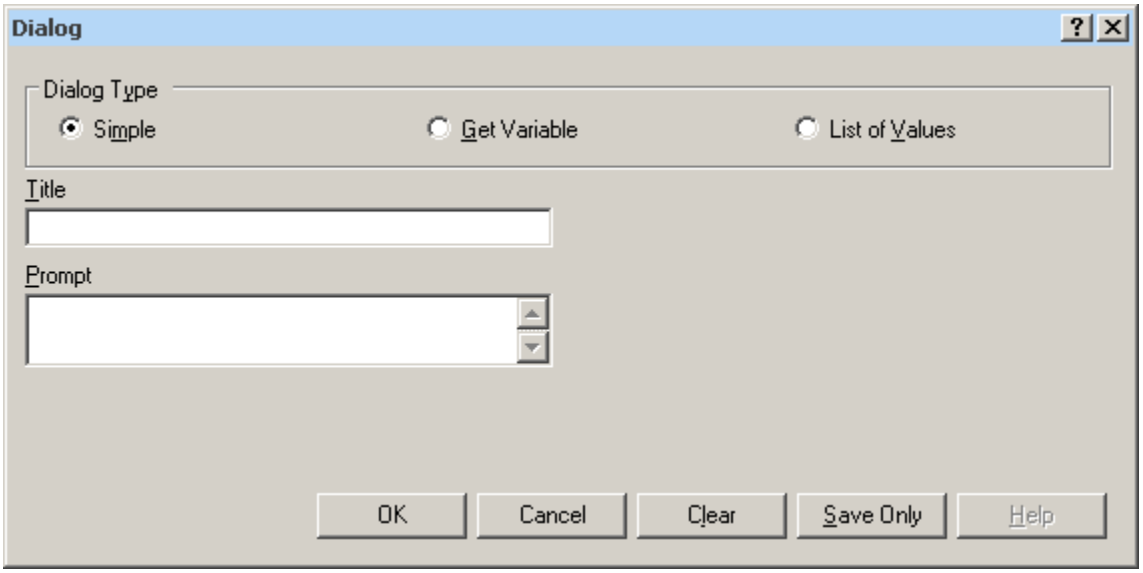


Figure 9.56: Dialog Command

DIALOG Type Simple

- **Title** holds the text of the heading title.
- **Prompt** contains the text to be used in the dialog as a message.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** closes the command generation window without saving or executing a command.
- **Clear** empties fields so information can be re-entered.
- **Save Only** saves the command in the Program Editor, but does not run the command.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

DIALOG Type Get Variable

This form of dialog displays a combo box of databases or form variables.

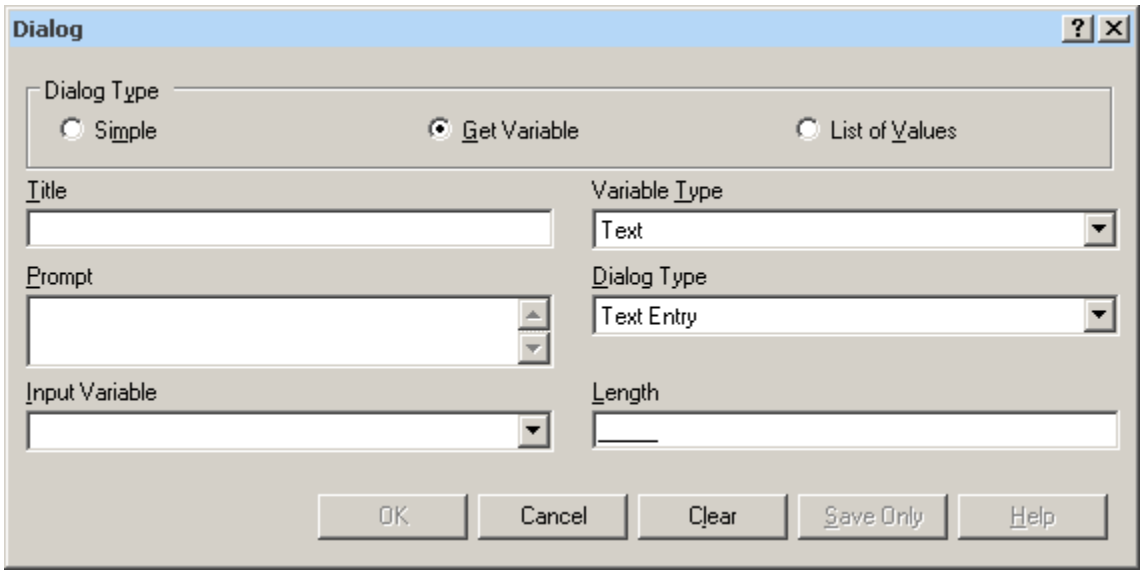


Figure 9.57: Dialog Type Combination Box

- **Title** holds the text of the heading title.
- **Prompt** contains the text to be used in the dialog as a message.
- **Input Variable** allows you to select a variable from the current data table.
- **Variable Type** allows you to select the dialog format to receive the value. Formats include Text, Number, Date, and Yes-No-Cancel. This field defaults to any previously assigned types. If the Variable Type is Text, the Dialog Type drop down allows you to select Text Entry, Multiple Choice, Variable List, Form List, Database List, File Open, and File Save. Database List options are not restricted to databases, but can include any file type. The Multiple Choice selection opens another set of dialog boxes to set up the choice selection.
- **Pattern** and/or **Length** create the pattern or size that allows input to follow when entered into the variable. Pattern options are based on Variable and Dialog Type selections. Number Type patterns consist of pound signs (#) and can contain one decimal place with a boundary. Date pattern options are DD-MM-YYYY, MM-DD-YYYY, or YYYY-MM-DD.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** closes the command generation window without saving or executing a command.
- **Clear** empties fields to re-enter information.

- **Save Only** saves the command in the Program Editor, but does not run the command.
- **Help** opens the Help topic associated with the module being used (Currently Disabled).

DIALOG Type List of Values

This dialog type displays a combo box of distinct values of the specified variable in the specified database.

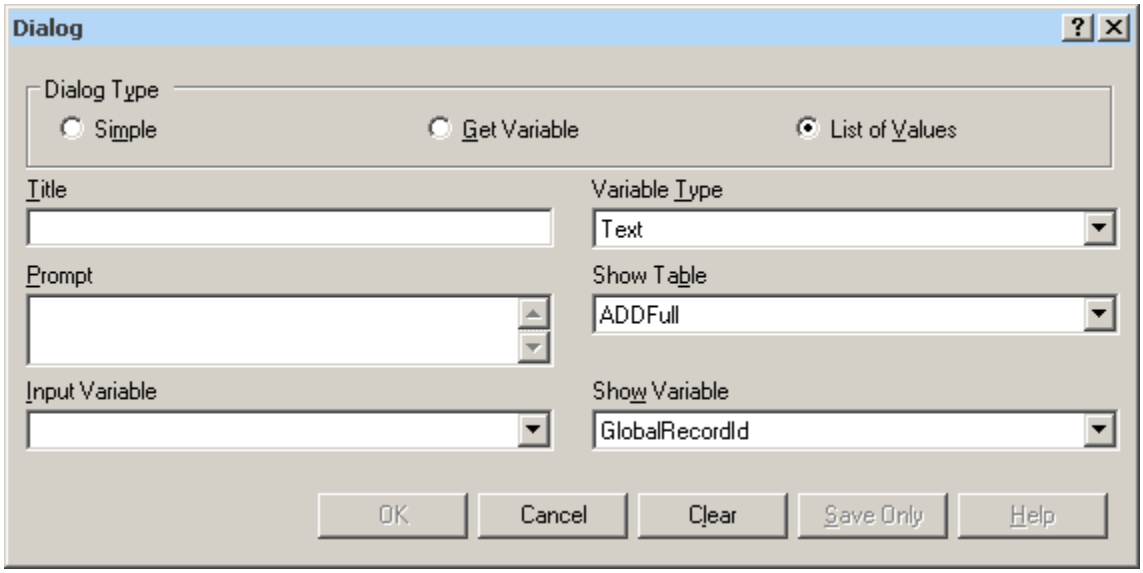


Figure 9.58: Dialog Type List of Values

- **Title** holds the text of the heading title.
- **Prompt** contains the text to be used in the dialog as a message.
- **Input Variable** allows you to select a variable from the current data table.
- **Variable Type** allows you to select the dialog format to receive the value. Options are Text, Number, Date, and Yes-No-Cancel.
- **Show Table** contains a list of available forms or tables in the current project.
- **Show Variable** contains a list of available fields that act as selection criteria for the input variable in the current project.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** closes the command generation window without saving or executing a command.

- **Clear** empties fields to re-enter information.
- **Save Only** saves the command in the Program Editor, but does not run the command.
- **Help** opens the Help topic associated with the module being used (Currently Disabled).

BEEP

The Beep command generates a sound when a function is performed.

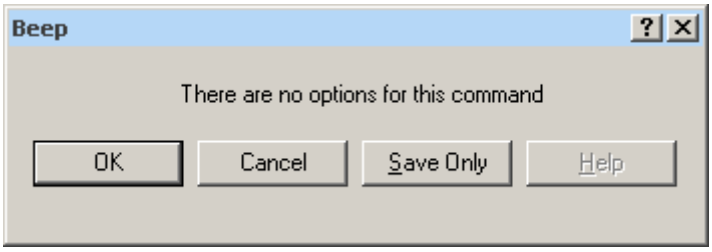


Figure 9.59: Beep Command Box

- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** closes the dialog box without saving or executing a command.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Help** opens the Help topic associated with the module being used (Currently Disabled).

QUIT

The Quit command closes the current data files and terminates the current program, closing Analysis.

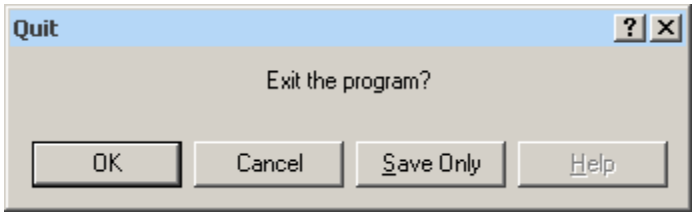


Figure 9.60: Quit Command Box

- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** closes the dialog box without saving or executing a command.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Help** opens the Help topic associated with the module being used (Currently Disabled).

Use IF Statements with Permanent or Global Variables

- If an IF statement condition depends on global or permanent variables, the value is computed immediately and only the true or false branch, as appropriate, is followed.
- If an IF statement condition depends on a field variable, neither branch is followed. However, computations within the branches are saved for execution, as appropriate, on each record. In the second case, non-computational commands (e.g., READ, SELECT, SORT, HEADER, and ROUTEOUT) are never executed.

SET

Description

This command provides various options that affect the performance and output of data in Classic Analysis. These settings are utilized whenever the Classic Analysis program is used.

Syntax

```
SET [<parameter> = <value>]
```

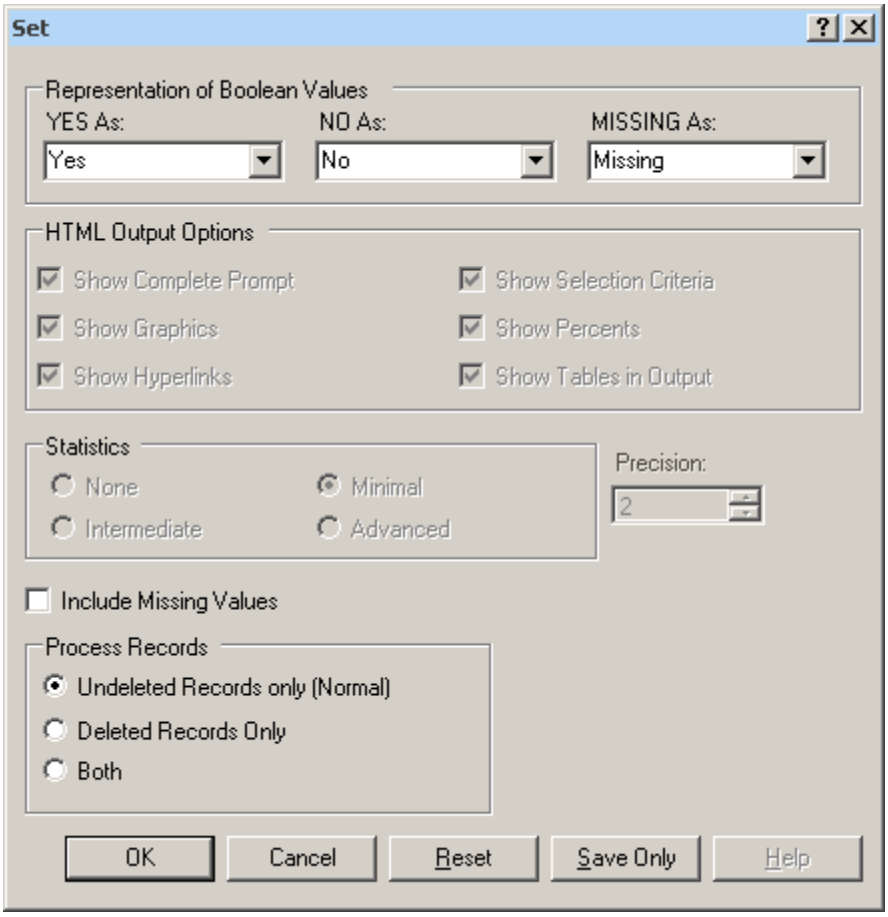


Figure 9.61: Set Options Window

- **Representation of Boolean Values** allows you to determine how values in the Epi Info 7 projects line list are displayed for Yes/No and Checkbox fields. Boolean Values are stored in the database according to the convention used by that database. Epi Info 7 converts the line listing displayed in analysis according to the selection in this menu.
- **HTML Output Options** are not available in this version of Epi Info 7.

- **Statistics** settings are not available in this version of Epi Info 7. The grayed-out display indicates analysis statistics are advanced. The None, Minimal and Intermediate settings will be available in a future release.
- **Precision** setting is not available in this version of Epi Info 7.
- **Include Missing Values** allows you to determine if you want missing values to be included if statistical calculations are made.
- **Process Records** selects how records marked for deletion (deleted) will be processed. The normal default is to process only the undeleted records.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Cancel** closes the dialog box without saving or executing a command.
- **Reset** clears (removes) any changes and returns all values to their default settings.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

How to Use Advanced Statistics

Use the REGRESS Command

The REGRESS command performs linear regression and contains support for automatic dummy variables and multiple interactions.
REGRESS can be used for simple linear regression (only one independent variable), for multiple linear regression (more than one independent variable), and for quantifying the relationship between two continuous variables (correlation). Regression is used when you want to predict one dependent variable from one or more independent variables.

Syntax

```
REGRESS <dependent variable> = <independent variable(s)> [NOINTERCEPT]
[OUTTABLE=<tablename>] [WEIGHTVAR=<weight variable>] [PVALUE=<PValue>]
```

Dialog Box

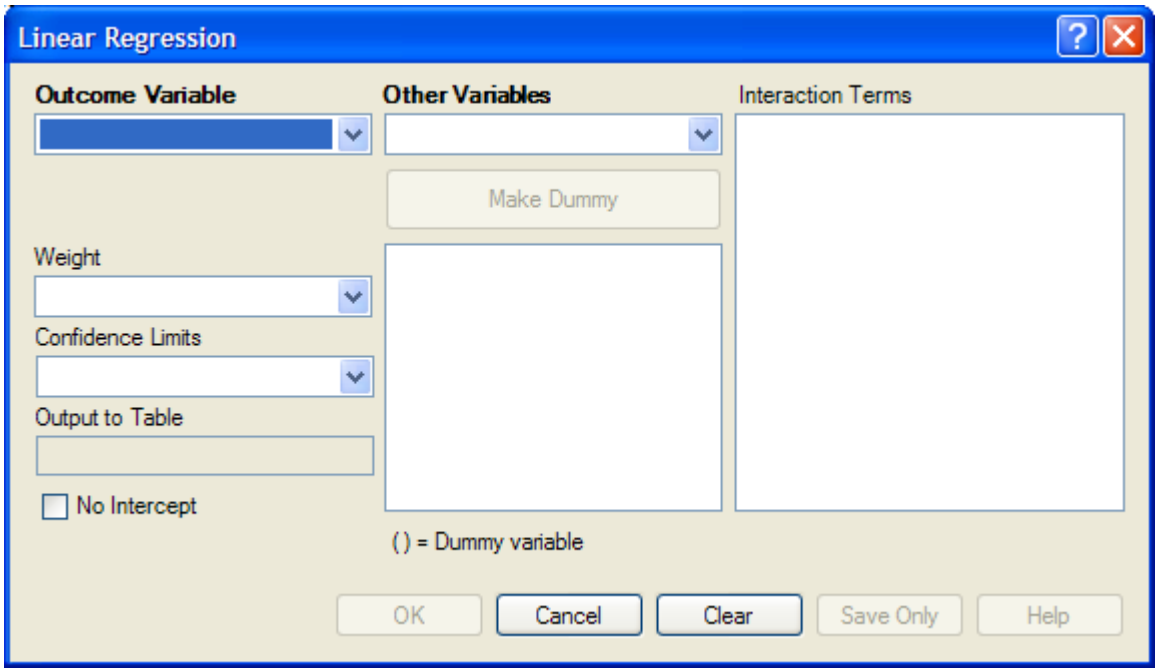


Figure 9.62: Linear Regression Window

- The **Outcome Variable** is the dependent variable for the regression.
- **Other Variables** appear in the predictor variables list.

- **Interaction Terms** are defined with the Make Interaction button. Make Interaction appears if two or more variables are selected from the Other Variables list box. If you click Make Interaction, the relationship populates the Interaction Terms list box.
- **Make Dummy** is activated if you select a predictor variable. If you select a numeric variable, it will be treated as discrete rather than continuous; the variable is enclosed in parentheses to indicate that dummy variables are being created. If a predictor variable enclosed in parentheses is selected in the list, the Make Dummy button changes to Make Continuous. Selecting it results in the variable being treated as continuous. If you select more than one predictor variable, the Make Dummy button changes to Make Interaction. Selecting it results in all possible combinations of the selected variables being added to the regression as interaction terms.
- A **Weight** variable may selected to use in weighted analyses.
- **Confidence Limits** specifies the probability level at which confidence limits are computed (default=.05).
- The **Output to Table** field identifies a table to receive output from the command. (Currently Disabled).
- If you select **No Intercept**, the regression is performed without a constant term, forcing the regression line through the origin.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

How to Use

1. From the Analysis Command Tree, use the READ command to open a **PRJ project file**. Select a **form** or **table**.
2. From the Analysis Command Tree, click **Advanced Statistics > Linear Regression**. The REGRESS dialog box opens.

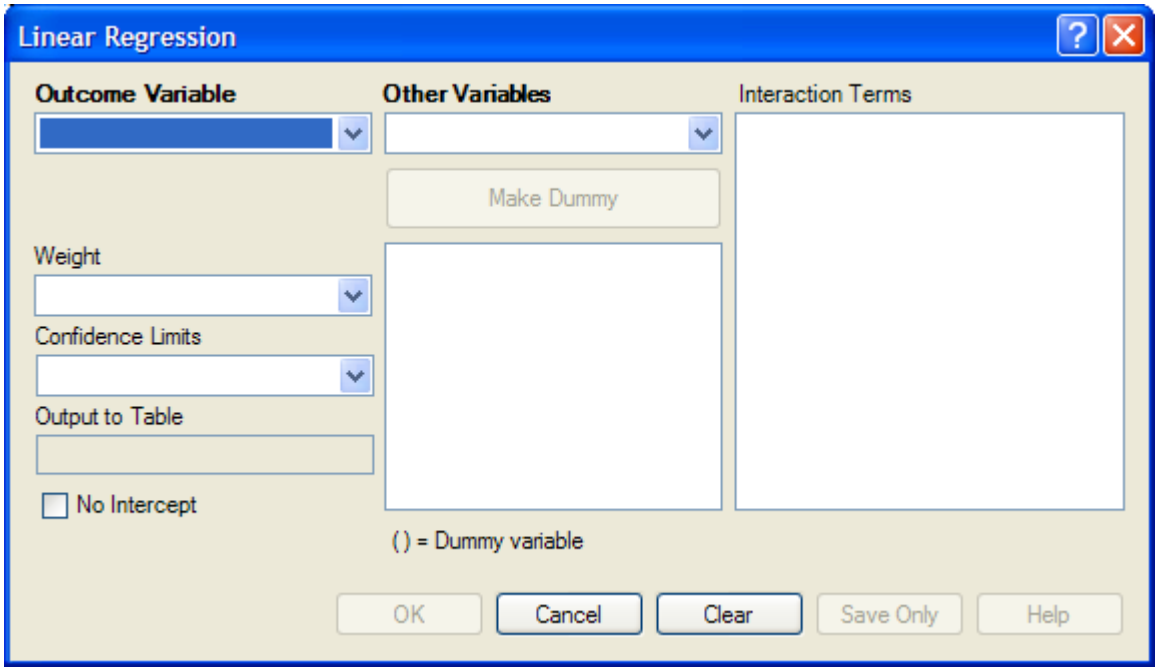


Figure 9.63: Linear Regression Window

3. From the Outcome Variable drop-down list, select a **variable** to be the dependent variable for regression.

4. From the Other Variables drop-down list, select the **variable(s)** to be the predictors.

   - If you select any predictor variables, Make Dummy will be activated. Selecting one will allow a numeric variable to become discrete rather than continuous; the variable is enclosed in parentheses to indicate that dummy variables are created. If a predictor variable enclosed in parentheses is selected, the Make Dummy button changes to Make Continuous. Selecting it makes the variable continuous. If you select more than one, the Make Dummy button changes to Make Interaction. Selecting it adds all possible selected combinations to the regression as interaction terms.

   - Make Interaction appears if you select two or more variables from the Other Variables list box. Interaction Terms are defined with the Make Interaction button. Click **Make Interaction.** The relationship populates the Interaction Terms list.

   - The Output to Table field identifies a data table to receive output from the command. Results can be sent to an output table for graphing. (Currently Disabled).

   - If No Intercept is selected, the regression is performed without a constant term, forcing the regression line through the origin.

5. Click **OK**. Results appear in the Output window.

Try It

1. Read in the project **Sample.PRJ.** Open **BabyBloodPressure**.
2. Click **Linear Regression**. The REGRESS dialog box opens.
3. From the Outcome Variable drop-down list, select **SystolicBlood**.
4. From the Other Variables drop-down list, select **AgeInDays**.
5. From the Other Variables drop-down list, select **Birthweight**.
6. Click **OK**. Results appear in the Output window.

Linear Regression

Variable	Coefficient	Std Error	F-test	P-Value
AgeInDays	5.888	0.680	74.9229	0.000002
Birthweight	0.126	0.034	13.3770	0.003281
CONSTANT	53.450	4.532	139.1042	0.000000

Correlation Coefficient: r^2= 0.88

Source	df	Sum of Squares	Mean Square	F-statistic
Regression	2	591.036	295.518	48.081
Residuals	13	79.902	6.146	
Total	15	670.938		

Use the LOGISTIC Command

The Logistic Regression command performs conditional or unconditional multivariate logistic regression with automatic dummy variables and support for multiple interactions. The dependent (Outcome) variable must have a Yes/No value. Records with missing values are excluded from the analyses.
Independent (Other Variables) can be numeric, categorical, or Yes/No variables. Missing is interpreted as missing, 0 is false, and any other response is true. Independent variables are controlled by the Include Missing setting. If Include Missing is used with missing values and true and false, dummy variables will be made automatically, which contribute Yes vs. Missing and No vs. Missing. Independent variables of text type are automatically turned into dummy variables, which compare each value relative to the value lowest in the sort order. Date or numeric type Independent variables are treated as continuous variables unless surrounded by parentheses in the command. If that occurs, they automatically turn into dummy variables which compare each value relative to the lowest value.

Syntax

```
LOGISTIC <dependent variable> = <independent variable(s)> [MATCHVAR=<match variable>]
[NOINTERCEPT] [OUTTABLE=<tablename>] [WEIGHTVAR=<weight variable>] [PVALUE=<PValue>]
```

Dialog Box

1. From the Analysis Command Tree, use the READ command to open a **PRJ project file**. Select a **form** or **table**.

2. From the Analysis Command Tree, click **Advanced Statistics > Logistic Regression**. The LOGISTIC dialog box opens.

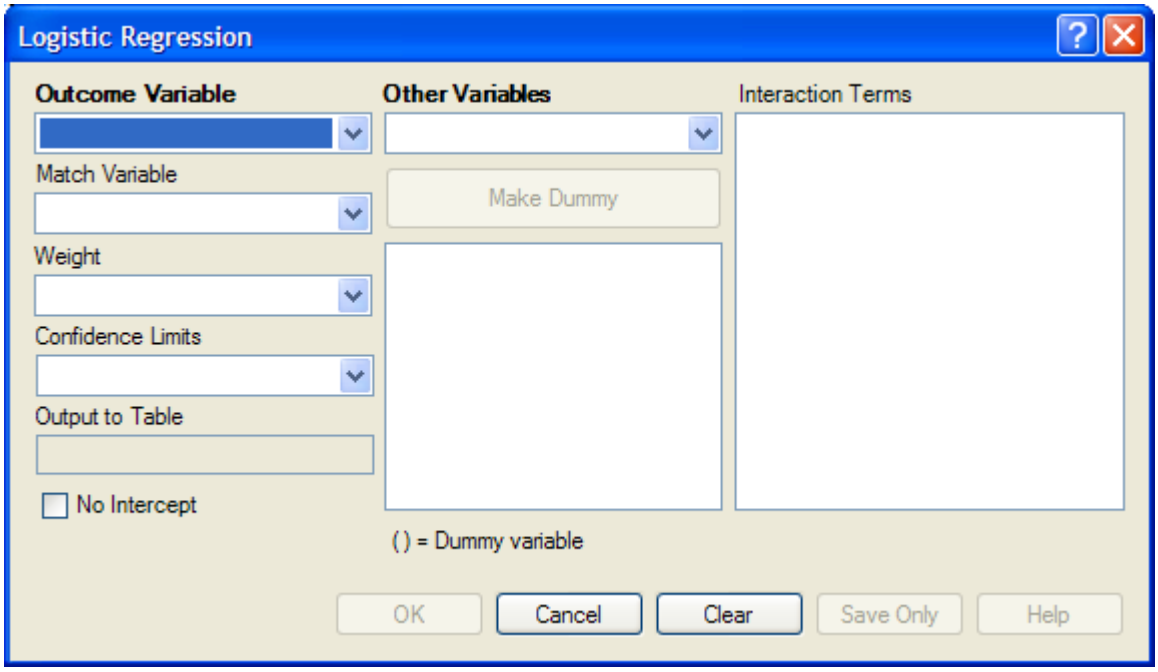


Figure 9.64: Logistic Regression Window

3. From the Outcome Variable drop-down list, select a **variable** to act as the dependent variable for regression.

4. From the Other Variables drop-down list, select the **variable(s)** to act as the predictors.

   - If you select any predictor variables, Make Dummy will be activated. Selecting one will allow a numeric variable to become discrete rather than continuous; the variable is enclosed in parentheses to indicate that dummy variables are created. If a predictor variable enclosed in parentheses is selected, the Make Dummy button changes to Make Continuous. Selecting it makes the variable continuous. If you select more than one, the Make Dummy button changes to Make Interaction. Selecting it adds all possible selected combinations to the regression as interaction terms.

   - **Make Interaction** appears if two or more variables are selected from the Other Variables list box. Interaction Terms are defined with the Make Interaction button. Click **Make Interaction.** The relationship populates the Interaction Terms list.

   - **Match Variable** identifies the variable indicating the group membership of each record.

   - The **Output to Table** field identifies a data table to receive output from the command. Results can be sent to an output table for graphing. (Currently Disabled).

- If **No Intercept** is selected, the regression is performed without a constant term forcing the regression line through the origin.

5. Click **OK**. Results appear in the Output window.

Try It

1. Read in the project **Sample.PRJ.** Open **Oswego**
2. Click **Logistic Regression**. The LOGISTIC dialog box opens.
3. From the Outcome Variable drop-down list, select **ILL**.
4. From the Other Variables drop-down list, select **BROWNBREAD**, **CABBAGESAL**, **WATER**, **MILK**, **CHOCOLATE**, and **VANILLA**.
5. Click **OK**. Results appear in the Output window.

Unconditional Logistic Regression

Term	Odds Ratio	95%	C.I.	Coefficient	S. E.	Z-Statistic	P-Value
BROWNBREAD (Yes/No)	1.7803	0.3932	8.0614	0.5768	0.7706	0.7485	0.4542
CABBAGESAL (Yes/No)	1.1342	0.2818	4.5647	0.1259	0.7104	0.1772	0.8593
WATER (Yes/No)	1.1122	0.2670	4.6326	0.1063	0.7280	0.1460	0.8839
MILK (Yes/No)	0.1342	0.0068	2.6635	-2.0086	1.5246	-1.3174	0.1877
CHOCOLATE (Yes/No)	1.0975	0.3024	3.9829	0.0930	0.6577	0.1415	0.8875
VANILLA (Yes/No)	26.0016	5.4707	123.5818	3.2582	0.7953	4.0968	0.0000
CONSTANT	*	*	*	-2.1277	0.9733	-2.1861	0.0288

Convergence:	Converged
Iterations:	5
Final -2*Log-Likelihood:	69.2504
Cases included:	74

Test	Statistic	D.F.	P-Value
Score	28.0180	6	0.0001
Likelihood Ratio	29.8484	6	0.0000

Use the KMSURVIVAL Command

The KMSURVIVAL command performs Kaplan-Meier (KM) Survival Analysis. The objective of this methodology is to estimate the probability of survival of a defined group at a designated time interval. KM uses a non-parametric survival function for a group of patients (their survival probability at time *t*) and does not make assumptions about the survival distribution.

What distinguishes survival analysis from most other statistical methods is the presence of "censoring" for incomplete observations. In a study following two different treatment regimens, analysis of the trial typically occurred well before all patients died. For those still alive at the time of analysis, the true survival time was known only to be greater than the time observed to date. These observations are called "censored." Two other sources of incomplete observation are patients "lost to follow-up," and the appearance of an event other than the event being studied.

Survival analysis requires censored and time variables, the units of time, and the groups being compared.

Syntax

```
KMSURVIVAL <TimeVar> = <GroupVar> *  <CensorVar> (<Value>) [TIMEUNIT="<TimeUnit>"]
[OUTTABLE=<TableName>] [GRAPHTYPE ="<GraphType>"] [WEIGHTVAR=<WeightVar>]
```

Dialog Box

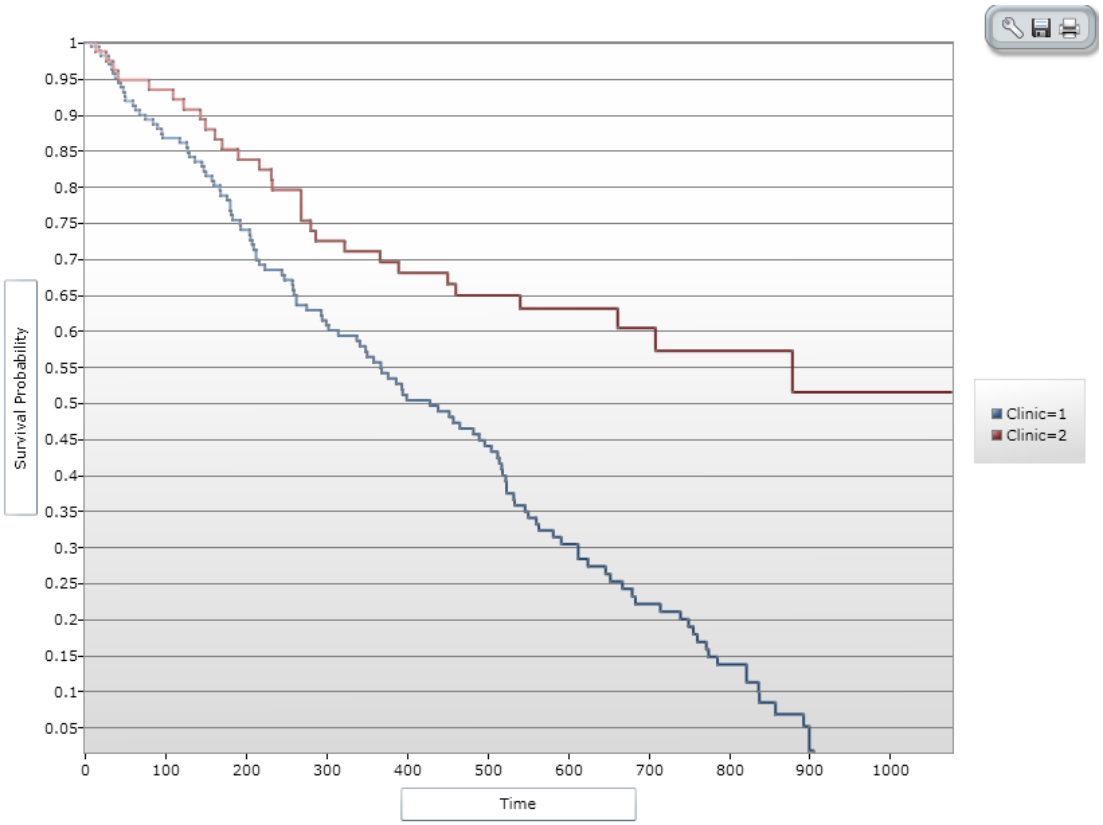
1. From the Analysis Command Tree, use the READ command to open a **PRJ project file**. Select a **form** or **table**.
2. From the Analysis Command Tree, click **Advanced Statistics > Kaplan-Meier Survival**. The Kaplan-Meier Survival dialog box opens.
3. From the Censored Variable drop-down list, select the **variable** that indicates whether the case is a failure or a censored case.
4. From the Value for Uncensored drop-down list, select the **value of the censored variable** that indicates a failure**.**
5. From the Time Variable drop-down list, select the **variable** that indicates at which time the failure or censorship occurred**.**
6. From the Test Group Variable drop-down list, select the **discrete variable** used to divide cases into groups.

7. Click **OK**. Results appear in the Output window.

- A Survival Probability graph is produced by default. Use the Graph Type drop-down list to select the **Log-Survival** graph type or **None** to view the results in a table.

Try It

1. Read in the **Sample.PRJ** project. Check the checkbox to Show Tables and scroll down to open the **Addicts** table.
2. Click **KAPLAN-MEIER SURVIVAL**. The Kaplan-Meier Survival dialog box opens.
3. From the Censored Variable drop-down list, select **Status**.
4. From the Value for Uncensored drop-down list, select **1**.
5. From the Time Variable drop-down list, select **Survival_Time_Days**.
6. From the Test Group Variable drop-down list, select **Clinic**.
7. Click **OK**. Results appear in the Output window.



Test	Statistic	D.F.	P-Value
Log-Rank	27.2477	1	0.0000
Wilcoxon	11.6304	1	0.0007

Figure 9.65: Results of Kaplan-Meier Example Survey

Use the COXPH Command

The COXPH command performs Cox Proportional Hazards survival analysis. This form of survival analysis relates covariates to failure through hazard ratios. A covariate with a hazard ratio less than one suggests improved survival for the level being compared with the reference level. COXPH output includes regression coefficients, test statistics with p-values, hazard ratios, and cumulative survival plots.

What distinguishes survival analysis from most other statistical methods is the presence of "censoring" for incomplete observations. In a study following two different treatment regimens, analysis of the trial typically occurred well before all patients died. For those still alive at the time of analysis, the true survival time is known only to be greater than the time observed to date. These observations are called "censored". Two other sources of incomplete observation are patients "lost to follow-up", and the appearance of an event other than the event being studied.
Survival analysis requires censored and time variables, the units of time, and the groups being compared.

Syntax

```
COXPH <time variable>= <covariate(s)>[: <time function>:]  *  <censor variable>
(<value>) [TIMEUNIT="<time unit>"] [OUTTABLE=<tablename>] [GRAPHTYPE="<graph type>"]
[WEIGHTVAR=<weight variable>] [STRATAVAR=<strata variable(s)>] [GRAPH=<graph
variable(s)>]
```

Dialog Box

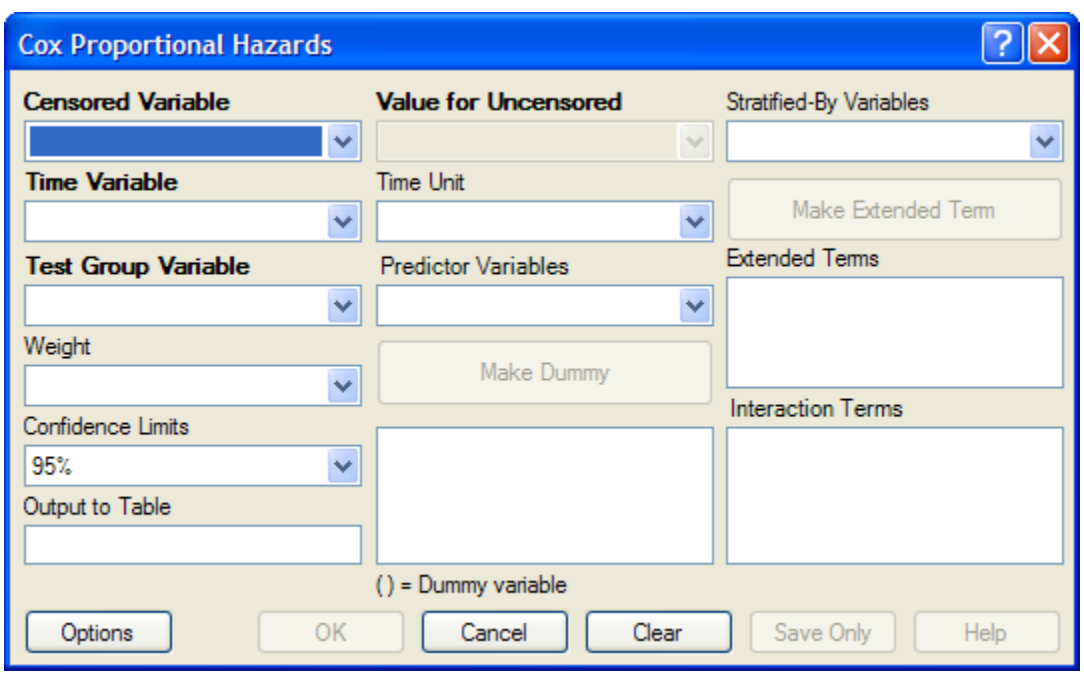


Figure 9.66: Cox Proportional Hazards Dialog Box

- The **Censored Variable** indicates whether the case is a failure or censored.
- The **Time Variable** indicates at which time the failure or censorship occurred.
- **Test Group Variable** is a discrete variable used to divide cases into groups for comparison. In the Cox model, there is no essential difference between the group variable and other predictor terms. For this reason, its use is optional.
- A **Weight** variable is selected for use in weighted analyses. The weight variable applies to all aggregation clauses.
- **Confidence Limits** indicate the probability level at which confidence limits should be computed (default=.05).
- The **Output to Table** field identifies a data table to receive output from the command. (Currently Disabled).
- **Value for Uncensored** is the value of the censored variable that indicates a failure.
- **Time Unit** is the unit in which the time variable is expressed.
- **Predictor Variables** populates the predictor variables list.
- If a predictor variable is selected, **Make Dummy** is active. If selected, a numeric variable is treated as discrete rather than continuous; the variable is enclosed in parentheses to indicate that dummy variables are being created. If you select a predictor variable enclosed in parentheses, the button changes to Make Continuous. Selecting it results in the variable being treated as continuous.
- **Stratify by** identifies the variable (if any) to be used to stratify data.
- **Options** opens the graph selection window and allows you to select variables for graphing and graph types.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Try It

1. From the Analysis Command Tree, use the READ command to open a **PRJ project file**. Select a **form** or **table**.

2. From the Analysis Command Tree, click **Advanced Statistics > Cox Proportional Hazards**. The Cox Proportional Hazards dialog box opens.
3. From the Censored Variable drop-down list, select the **variable** that indicates whether the case is a failure or a censored case.
4. From the Value for Uncensored drop-down list, select the **value** of the censored variable that indicates a failure.
5. From the Time Variable drop-down list, select the **variable** that indicates at which time the failure or censorship occurred.
6. From the Time Unit drop-down list, select the **unit** in which the time variable is expressed, if needed.
7. From the Test Group Variable drop-down list, select the **discrete variable** used to divide cases into groups.
   - In the Cox model, there is no essential difference between the group variable and other predictor terms. For this reason, using the group variable is optional.
8. From the Other Variables drop-down list, select the **predictor variables.**
   - If you select any predictor variables, Make Dummy will be activated. Selecting will allow a numeric variable to become discrete rather than continuous; the variable is enclosed in parentheses to indicate that dummy variables are created. If a predictor variable enclosed in parentheses is selected in the list, the Make Dummy button changes to Make Continuous. Selecting it makes the variable continuous. If you select more than one, the Make Dummy button changes to Make Interaction. Selecting it results in all possible combinations of the selected variables being added to the regression as interaction terms.
9. Click **Graph Options**. The Cox Graph Options dialog box opens.
10. From the Plot Variables drop-down list, select a **graph type.**
11. From the list box, select a **variable(s) to graph**.
12. Clear the **Customize Graph** checkbox.
13. Click **OK**. The Cox Proportional Hazards dialog box opens.
14. Click **OK**. Results appear in the Output window.

Try It

1. Read in the **Sample.PRJ** project. Open the **Addicts** table.
2. Click **Cox Proportional Hazards**. The Cox Proportional Hazards dialog box opens.
3. From the Censored Variable drop-down list, select **Status**.
4. From the Value for Uncensored drop-down list, select **1**.

5. From the Time Variable drop-down list, select **Survival_Time_Days**.
6. From the Test Group Variable drop-down list, select **Clinic**.
7. From the Predictor Variables drop-down list, select **Methadone_dose__mg_day** and **Prison_Record**.
8. Click **Options**. The Cox Graph Options dialog box opens.
9. From the Plot Variables drop-down list, select **Survival Observed**.
10. From the list box, select **Clinic**.
11. Clear the **Customize Graph** checkbox.
12. Click **OK**. The Cox Proportional Hazards dialog box opens.
13. Click **OK**. Results appear in the Output window.

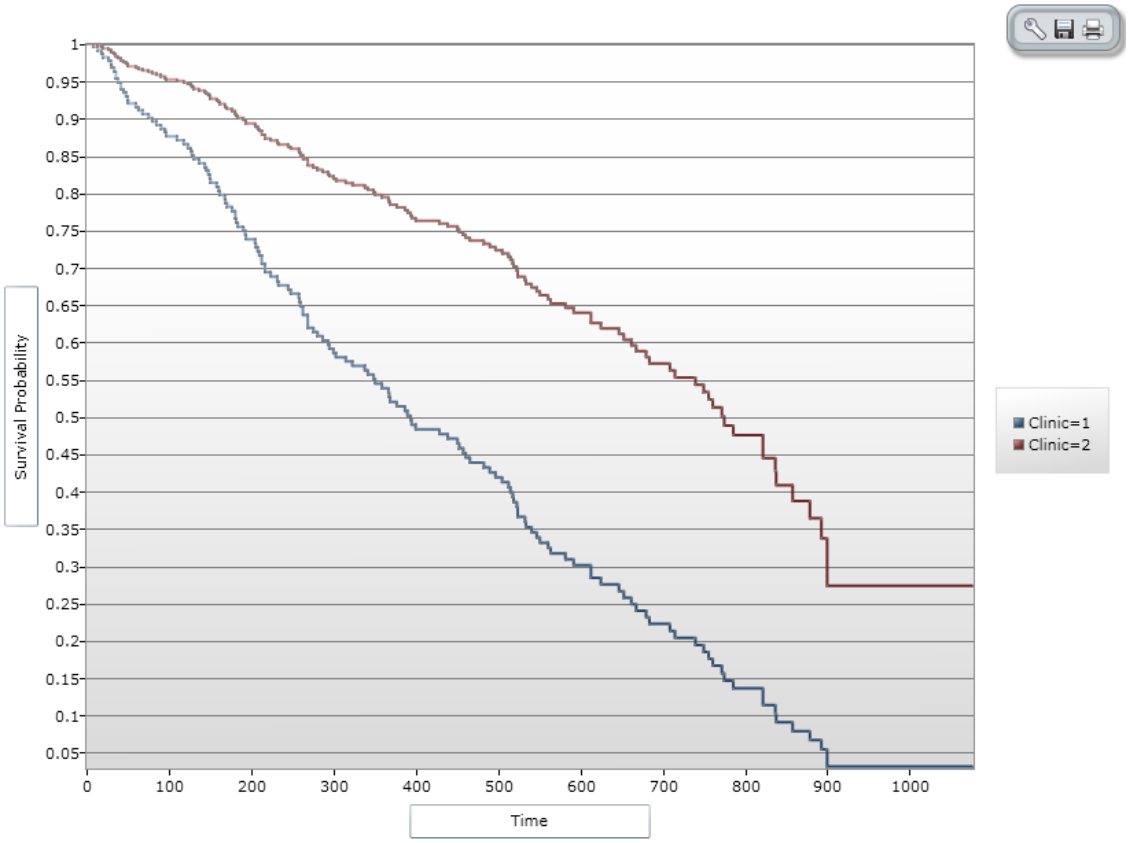


Figure 9.67: Cox Proportional Hazards Graph Result

Cox Proportional Hazards

Term	Hazard Ratio	95%	C.I.	Coefficient	S. E.	Z-Statistic	P-Value
Clinic (2/1)	0.3724	0.2460	0.5636	-0.9879	0.2114	-4.6719	0.0000
Methadone_dose__mg_day_	0.9656	0.9535	0.9778	-0.0350	0.0064	-5.4680	0.0000
Prison_Record	1.3856	0.9970	1.9257	0.3261	0.1679	1.9419	0.0521

Convergence: Converged
Iterations: 5
-2 * Log-Likelihood: 1347.2015

Test	Statistic	D.F.	P-Value
Score	54.9131	3	0.0000
Likelihood Ratio	62.6726	3	0.0000

Figure 9.68: Cox Proportional Hazards Statistical Results

Complex Sample Frequencies, Tables, and Means

The Frequencies (FREQ), Tables (TABLES), and Means (MEANS) commands in the Classic Analysis program perform statistical calculations that assume the data comes from simple random (or unbiased systematic) samples. More complicated sampling strategies are used in many survey applications. These may involve sampling features (i.e., stratification, cluster sampling, and the use of unequal sampling fractions). Surveys that include some form of complex sampling include the coverage surveys of the WHO Expanded Program on Immunization (EPI) (Lemeshow and Robinson, 1985) and CDC's Behavioral Risk Factor Surveillance System (Marks et al., 1985).
The CSAMPLE functions compute proportions or means with standard errors and confidence limits for studies where the data did not come from a simple random sample. If tables with two dimensions are requested, the odds ratio, risk ratio, and risk difference are also calculated.

Data from complex sample designs should be analyzed with methods that account for the sampling design. In the past, easy-to-use programs were not available for analysis of such data. CSAMPLE provides these facilities. It can form the basis of a complete survey system with an understanding of sampling design and analysis.

Complex Sample Frequencies

Dialog Box

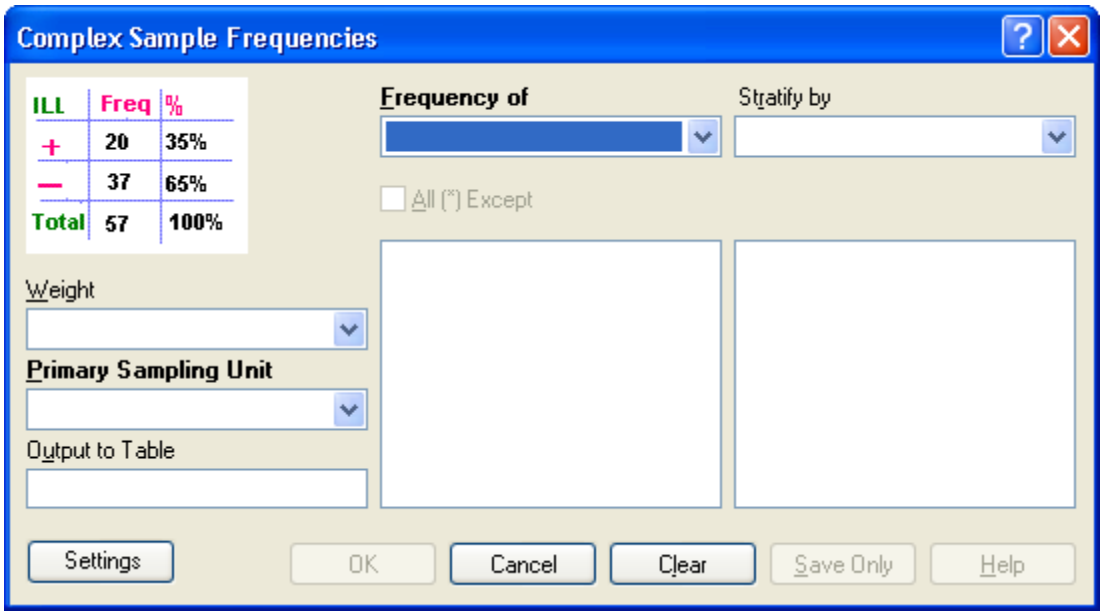


Figure 9.69: Complex Sample Frequencies Dialog Box

- A **Weight** variable is selected for use in weighted analyses.
- The **Output to Table** field identifies a data table to receive output from the command.
- **Frequency of** identifies the variable(s) whose frequency is computed.
- **All (*) Except** indicates that all the variables except those selected will have frequencies computed.
- **Stratify by** identifies the variable to be used to stratify or group the frequency data.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Complex Sample Means

Dialog Box

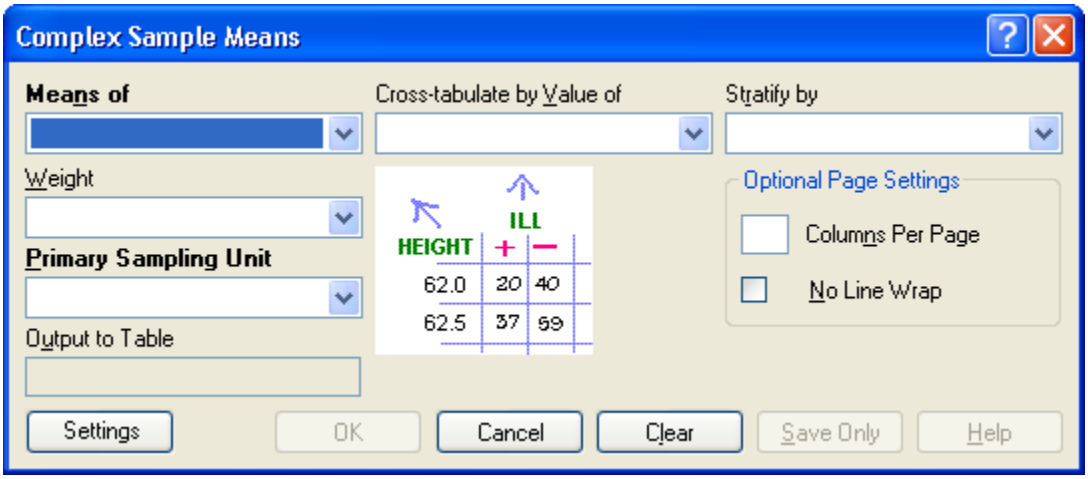


Figure 9.70: Complex Sample Means Dialog Box

- **Means of** identifies the variable whose mean is to be computed
- A **Weight** variable is selected for use in weighted analyses.
- The **Output to Table** field identifies a data table to receive output from the command. (Currently Disabled).
- **Cross-Tabulate by Value of** identifies the variable to be used to cross-tabulate the main variable.
- **Stratify by** identifies the variable to be used to stratify or group the frequency data.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).

Complex Sample Tables

Dialog Box

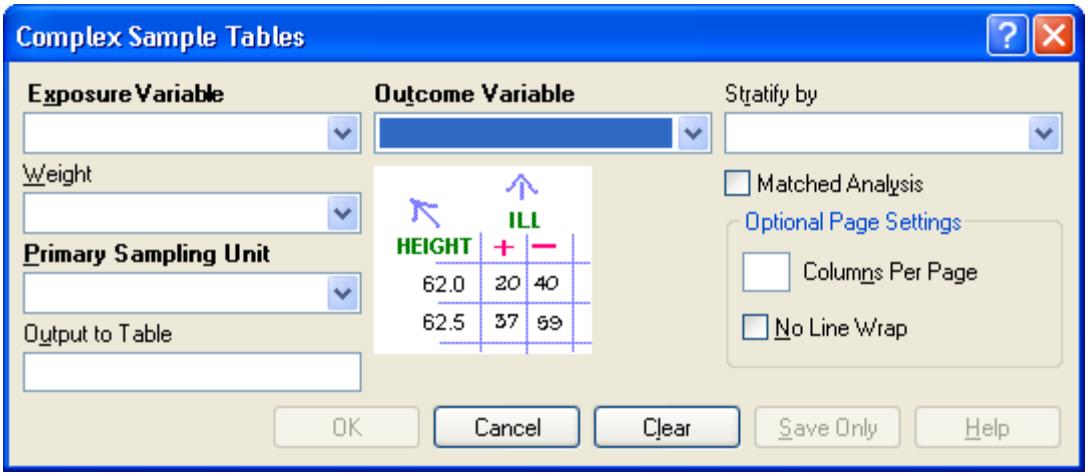


Figure 9.71: Complex Sample Tables Dialog Box

- **Exposure Variable** identifies the variable that will appear on the horizontal axis of the table. It is considered to be the risk factor (or * for all variables).
- A **Weight** variable is selected for use in weighted analyses.
- The **Output to Table** field identifies a data table to receive output from the command.
- **Outcome Variable** identifies the variable that will appear on the vertical axis of the table.
- **Stratify by** identifies the variable to be used to stratify or group the frequency data.
- **OK** accepts the current settings and data, and subsequently closes the form or window.
- **Save Only** saves the created code to the Program Editor, but does not run the code.
- **Cancel** closes the dialog box without saving or executing a command.
- **Clear** empties the fields so information can be re-entered.
- **Help** opens the Help topic associated with the module being used. (Currently Disabled).