Chapter 130

Macros

Introduction

This software has an interactive (point and click) user interface which makes it easy to learn and use. At times, however, it is necessary to repeat the same steps over and over. When this occurs, a batch system becomes more desirable. This chapter documents a batch language that lets you create a macro (script or program) and then run that macro. With the click of a single button, you can have the program run a series of procedures.

We begin with a discussion of how to create, modify, and run a macro. Next, we list all of the macro commands and their functions.

Macro Command Center

This section describes how to create, edit, and run a macro. This is all accomplished from the Macro Command Center window. You can load this window by selecting *Tools* > *Macros* > *Home...* from the menu.

We will now describe each of the objects on the Macro Command Center window.

Active Macro

This box displays the name of the currently selected macro file. A preview of this macro is displayed in the *Preview of Active Macro* box. You can change this name by typing over it or by selecting a different macro from the *Existing Macros* box.

Existing Macros

This box displays a list of all existing macros. Click on a macro in this list to make it the active macro, the macro that is used when one of the control buttons to the right is pressed. Double-clicking a macro causes that macro to open in the Windows Notepad program for editing.

The macros are stored in the *MACROS* subdirectory of the *NCSS* folder. They always have the file extension ***.ncm***. You can change the location where **NCSS** stores and loads macros using the *Folders* tab on the **NCSS** *System Options* window.

Preview of Active Macro

This box displays the beginning of the macro that is in the Active Macro box.

Record Macro

Pressing this button causes the commands you issue to be written to the macro file named in the *Active Macro* box. The Macro Command Center window will be replaced by a small *Macro* window that lets you stop recording the macro. When the recording starts, if there is a previous macro file with the same name as that in the Active Macro Name box, the previous macro file is erased.

The **NCSS** macro recorder does not record every keystroke and click that you make. Instead, it records major operations. For example, suppose you want to include running a t-test in your macro. You would load the t-test, change some options, and run it. The macro recorder saves a copy of the t-test settings as a file and writes a single command line to the macro file that references this settings file. All of your settings are included in the file—there is no reference in the macro to the individual settings changes. This makes the macro much smaller and easier to modify.

Not all macro commands are written out when recording a macro. Commands related to editing cells in a dataset or changing column information are not recorded. Only high-level data functions like open, import, and save are recorded. You should review the macro after recording to make sure it is complete for your needs.

Edit Macro

Pressing this button causes the Active Macro to be loaded in the Windows Notepad program. This program lets you modify the macro and then save your changes.

Play Macro

Pressing this button causes the Active Macro to be run. Once the macro is finished, the Macro Command Center window will close, and you can view the results of running your macro.

Delete Macro

Pressing this button causes the Active Macro to be deleted. The macro file is actually moved to the Recycle Bin from which it can be rescued if you decide it shouldn't have been deleted.

Close

Pressing this button closes the Macro Command Center window.

Set Menu Macro

A play macro menu item is available from the *Tools* menu. Clicking this play menu item causes the designated macro to be run. The macro that is associated with the macro menu is controlled by this section of the Macro Command center. To change the macro that is associated with the play macro menu item, simply select the desired macro from the *Existing Macros* list and then click the *Menu Macro* button. This will associate the active macro with the menu item. This association will remain even after you exit the program.

Syntax of a Macro Command Line

NCSS macros are line based. That is, each macro command expression is written on a separate line. The basic structure of a line is that it begins with a *command* followed by one or more options or parameters of the command, called *arguments*. For example, the following macro opens a dataset and runs the Descriptive Statistics procedure on the first five columns in that dataset.

```
DataOpen "C:\Users\User\Documents\NCSS\Sample.NCSS"
LoadProc DescStat
Option DescStat 1 "1:5"
RunProc DescStat
```

In this example, DataOpen, LoadProc, Option, and RunProc are commands, while "C:\Users\User\Documents\NCSS\Sample.NCSS", DescStat, 1, and "1:5" are arguments.

Comment Lines

It is often useful to add comment lines to a macro to make it easier to understand later. Comment lines begin with single quotes. When the macro processor encounters a single quote at the beginning of a line, the rest of the line is ignored. Single quotes occurring at a location other than the beginning of a line are treated as text.

Blank lines may also be added to a macro to improve readability. These are also ignored.

Macro Constants and Macro Variables

As stated above, macro commands lines consist of keywords followed by arguments. These arguments are either constants, such as "1" or "DescStat", or macro variables. These will be discussed next.

Macro Constants

Macro constants fixed values. There are two types of macro constants: text and numeric.

Numeric Constants

Numeric constants are numbers. They may be whole or decimal numbers. They may be positive or negative. They may be enclosed in double quotes, although this is not necessary. When the macro processor expects a number but receives a text value, it sets the numeric value to zero.

Examples of numeric constants are

```
1, 3.14159, and 0.
```

Text Constants

Text constants are usually enclosed in double quotes. If a constant is a single word (made of letters and digits with no blanks or special characters), the double quotes are not necessary.

Examples of text constants are

```
Apple, "Apple Pie", and "C:\Users\User\Documents\NCSS"
```

Macro Variables

Macro Variables are used to store temporary values for use in macro command lines. Some examples of assigning values to macro variables are

```
A# = 4
B# = 4 + 3
File$ = "C:\Users\User\Documents\NCSS\Data\ABC.NCSS"
F$ = "4" & "5"
```

In these examples, A#, B#, File$, and F$ are macro variables. The assigned values for each of the variables are 4, 7, “C:\Users\User\Documents\NCSS\Data\ABC.NCSS”, and 45, respectively.

There are two types of macro variables: text and numeric.

Text Macro Variables

Text macro variables are used to hold text values. The rules for naming them are that the names can contain only letters and numbers (no spaces or special characters) and they must end with a '$'. The case of the letters is ignored (so “A$” is used interchangeably with “a$”).

Examples of text macro variable names are

```
A$
Apple$
FileName$
```

Numeric Macro Variables

Numeric variables are used to hold numeric values. The rules for naming them are that the names can contain only letters and numbers (no spaces or special characters), and they must end with a “#”. The case of the letters is ignored (so “A#” is used interchangeably with “a#”).

Examples of numeric macro variable names are

```
A#
Apple#
NRows#
```

Assigning Values to Macro Variables

One type of macro variable expression is that of assigning a value to a variable. The basic syntax for this type of expression is

{variable} = {value}

where the {variable} is text or numeric and {value} is a macro variable or (text or numeric) macro constant. If a text value contains spaces or special characters, it must be enclosed in double quotes.

A text value can be is assigned to a numeric macro variable. In this case, the text value is converted to a number. If it cannot be converted to a number (e.g., it is a letter), the numeric macro variable is set to zero.

The following are some examples of valid assignment expressions:

```
A# = 4
X$ = John
File$ = "C:\Users\User\Documents\NCSS\Data\ABC.NCSS"
X$ = File$
X$ = A#
A# = F$
F$ = 4
```

Macro Variable Combination Expressions

Macro variables can be combined using simple mathematical expressions. The basic syntax for this type of expression is

{variable} = {value} {operator} {value}

The available operators are + (add), - (subtract), * (multiply), / (divide), and & (concatenate).

If a text value is involved in a mathematical expression, it is converted to a numeric value before the mathematical expression is evaluated. If it cannot be converted to a number, the text value is set to the numeric value of zero.

The following are some examples of valid assignment expressions:

Expression	**Result**
A# = 4 + 3	A# = 7
B# = A# * 2	B# = 14
C$ = "C:\Pgm\"	
D$ = C$ & "ABC.NCSS"	D$ = "C:\Pgm\ABC.NCSS"
E# = C$ * 4	E# = 0
F$ = "4" + "5"	F$ = "9"
F$ = "4" & "5"	F$ = "45"
A# = 1	
A# = A# + 1	A# = 2

Macro variable assignments are used while the macro is running but are not saved when the macro has completed.

Displaying Macro Variables

The value of one or more macro variables may be displayed on the output using the *Print* or *Heading* commands.

List of Macro Variable Display commands:

Print
Heading

Print

This command outputs the requested values to the printout.

The syntax of this command is

Print {p1} {p2} {p3} ...

where

{p1} {p2} ... are assigned macro variables or constants.

Following are some examples of *Print* commands:

Command	**Printed Result**
Print "Hi World"	Hi World
I# = 1	
J$ = "C:\NCSS\Data\ABC"	
F$ = J$ & I#	
F$ = F$ & ".NCSS"	
Print "File=" F$	File=C:\NCSS\Data\ABC1.NCSS
Print "1" "2" "3" "4"	1 2 3 4

Heading

This command adds a line to the page heading that is shown at the top of each page.

The syntax of this command is

Heading {h1}

where

{h1} is an assigned macro variable or constant.

The following are some examples of valid *Heading* commands:

Command	Heading
Heading "Hi World"	Hi World
F$ = "Heart Study"	
Heading F$	Heart Study

Logic and Control Commands

The lines in a macro are processed in succession. These commands allow you to alter the order in which macro lines are processed, allow user-input, or end the program.

List of Logic and Control Commands:

Flag
GoTo
If
Input
End
SendKeys

Flag

A flag is a reference point in the program. The *Goto* command sends macro line control to a specific flag. A flag is made up of letters and numbers (no spaces) followed by a colon.

The following are some examples of valid flags:

Examples
Flag1:
A:
Loop1:

More extensive examples are shown in the description of the *If* statement (below).

GoTo

This command transfers macro processing to the next statement after a flag.

The syntax of this command is

GoTo {p1}

where

{p1} is a text variable or text constant.

The following are some examples of valid *GoTo* commands:

Examples
GOTO Flag1
or
F$ = "Flag1"
GOTO F$

If

This command transfers macro processing to the next statement after a flag if a condition is met. The syntax of this command is

If {p1} {logic} {p2} GoTo {p3}

where

{p1} is a variable or constant.

{p2} is a variable or constant.

{p3} is a flag.

{logic} is a logic operator. Possible logic operators are =, <, >, <=, >=, and <>.

The following are some examples of valid *If* commands:

Examples
If x1# > 5 GoTo flag1
If y$ = "A" GoTo flag2
If y$ <> "A" GoTo flag3

Input

This command stops macro execution, displays a message window, and waits for a value to be input. This value is then stored in the indicated macro variable.

The syntax of this command is

Input {variable} {prompt} {default}

where

{variable} is the name of the variable (text or numeric) to receive the value that is input.

{prompt} is the text phrase that is shown on the input window.

{default} is the default value for the input.

Following is an example of this command:

Example
Input A# "Enter the number of items" 1

End

This command closes the **NCSS** system.

The syntax of this command is

End

Following is an example of this command:

Example
End

SendKeys

This command sends one or more keystrokes to the program as if you had typed them in from the keyboard. This facility allows you to create macros to accomplish almost anything you can do interactively within the program.

To use this, run the program from the keyboard, noting exactly which keys are pressed. Then, type the appropriate commands into the SendKeys text. Note that spaces are treated as characters, so “{down} {tab}” is different from “{down}{tab}”.

The syntax of this command is

SendKeys {value}

where *{value}* is a text constant or variable.

Remarks

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, use "A" for value. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, use "ABC" for string.

The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses ( ) have special meanings to SendKeys. To specify one of these characters, enclose it within braces ({}). For example, to specify the plus sign, use {+}. Brackets ([ ]) have no special meaning to SendKeys, but you must enclose them in braces. To specify brace characters, use {{} and {}}.

To specify characters that are not displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use the following codes:

Key	Code
Backspace	{bs}
Break	{break}
Caps Lock	{capslock}
Delete	{delete}
Down Arrow	{down}
End	{end}
Enter	{enter}
Esc	{esc}
Home	{home}
Insert	{insert}
Left Arrow	{left}
Num Lock	{numlock}
Page Down	{pgdn}
Page Up	{pgup}
Right Arrow	{right}
Tab	{tab}
Up Arrow	{up}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following codes:

Key	Code
Shift	+
Ctrl	^
Alt	%

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use "+(EC)". To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use "+EC".

To specify repeating keys, use the form {key number}. You must put a space between key and number. For example, {LEFT 4} means press the LEFT ARROW key 4 times; {h 8} means press H 8 times.

Data Window Note: When a macro is run the usual beginning location on the screen is the new page icon (just below File) in the upper left of the screen. A single tab may be entered in the macro to go to the upper left cell position on the data table.

Examples	Action
SendKeys "ABC {enter}"	(Types ABC and then enter)
SendKeys "{enter right}"	(Enter and then down arrow)
SendKeys "%H{down 2}{enter}"	(Activates the Serial Numbers from the Help menus)
SendKeys "{right}"	(Moves to the right one cell)

Window Position Commands

These commands allow the user to position or hide windows while the macro is running.

List of Window Commands:

WindowLeft
WindowTop

WindowLeft

This command sets the position of the left edge of the data, procedure, and output windows. This allows you to effectively hide these windows while a macro is running.

The syntax of this command is

WindowLeft {value}

where

{value} is the value of the left edge in thousandths of an inch.

The following are examples of this command:

Examples	Action
WindowLeft 0	(positions the left edge to zero)
WindowLeft 10000	(positions the left edge ten inches to the right)
WindowLeft -10000	(positions the left edge ten inches to the left)

WindowTop

This command sets the position of the top of the data, procedure, and output windows. This allows you to effectively hide these windows while a macro is running.

The syntax of this command is

WindowTop {value}

where

{value} is the value of the left edge in thousandths of an inch.

The following are examples of this command:

Examples	Action
WindowTop 0	(positions the top to zero)
WindowTop -10000	(positions the top ten inches up)

Dataset Commands

The following commands open, close, and modify an **NCSS** dataset.

List of Dataset Commands:

DataOpen
DataNew
DataSave
Import
Export
ResizeRowsCols
SortBy
GetCell
SetCell
NumRows
NumCols
ColFromList
GetMaxRows
GetMaxCols
ColName
ColLabel
ColDataType
ColFormat
ColValueLabel
ColValueOrder
ColTrans
ColNote
ColGroupBy
SetGroupByCols
SetGroupByActive
SetFilter
SetFilterActive
SetFilterHideRows
SetTransAutoRecalc
RunTrans

DataOpen

This command opens the data file given by *{file name}.*

The syntax of this command is

DataOpen {file name}

where

{file name} a text variable or text phrase enclosed in double quotes that gives the name of the **NCSS** data file to open.

The following are some examples of macro snippets that use this command:

Examples

```
DataOpen "C:\Users\User\Documents\NCSS\Sample.NCSS"

or

F1$="C:\Users\User\Documents\NCSS\"
F2$="Sample.NCSS"
F$=F1$ & F2$
DataOpen F$
```

DataNew

This command creates a new, untitled dataset.

The syntax of this command is

DataNew

Following is an example of this command:

Example

```
DataNew
```

DataSave

This command saves the current dataset to a file.

The syntax of this command is

DataSave {file name} {row save type} {column save type}

where

{file name}	is a text variable or text phrase enclosed between double quotes. Note that the extension of the file name must be ".NCSS". If a file with name already exists, it will be replaced.
{row save type}	a variable indicating how the rows are to be saved (0 = save all rows, 1 = only save rows that pass the filter (ignored if no filter active), 2 = save only the user-specified rows). This parameter is optional (if you wish to save a subset of columns but all rows, then you should enter a 0 here). If left out, all rows will be saved. When this option is set to 2, the proper syntax to specify the rows to save is "2;*<row list and/or range>*" (including the quotes) (Example: "2;1-25,30,50-70" would save rows 1 through 25, row 30, and rows 50 through 70).
{column save type}	a variable indicating how the columns are to be saved (0 = save all columns, 2 = save only the user-specified columns). This parameter is optional. If left out, all columns will be saved. When this option is set to 2, the proper syntax to specify the columns to save is "2;*<column list and/or range>*" (including the quotes). Column names or numbers can be entered. (Example: "2;C1-C3,C5,C6-C9" would save columns C1 through C3, column C5, and columns C6-C9. You could get the same result by entering "2;1-3,5,6-9").

The following are some examples of macro snippets that use this command:

Examples

```
DataSave "C:\Users\User\Documents\NCSS\Sample.NCSS"          (Saves all rows and all columns)
DataSave "C:\Users\User\Documents\NCSS\Sample.NCSS" 1        (Saves only non-filtered rows and all
                                                             columns)
DataSave "C:\Users\User\Documents\NCSS\Sample.NCSS" "2;1-30"       (Saves only rows 1 through 30
                                                                   and all columns)
DataSave "C:\Users\User\Documents\NCSS\Sample.NCSS" 0 "2;1-5"      (Saves all rows and only
                                                                   columns 1 through 5)
DataSave "C:\Users\User\Documents\NCSS\Sample.NCSS" "2;1-30" "2;C1-C5,C6,C12"
                                                        (Saves only rows 1 through 30 and only
                                                         columns C1 through C5, C6, and C12)

or

F1$="C:\Users\User\Documents\NCSS\"
F2$="Sample.NCSS"
F$=F1$ & F2$
DataSave F$
```

Import

This command imports a file of a different format and translates it into the **NCSS** format. Importing data is discussed in detail in the Importing Data chapter. We refer you to that chapter for the details of data importing.

The syntax of this command is

Import {type} {file name} {columns} {data types} {p1} {p2} {p3}

where

{type} a text variable or text phrase enclosed in double quotes that gives the type of data file to be imported. Some example file types are NCSS_S0, NCSS_S0z, Access, Excel_XLS, Excel_XLSX, ASCII_Delimited, and ASCII_CSV. See the Importing Data chapter for a complete list of importable file types.

{file name} a text variable or text phrase enclosed in double quotes that gives the name of the data file to be imported.

{columns} a list of column numbers to import. Separate items in the list with commas (Example: “1, 2, 5, 9”). Specify a range of values with the syntax “{Beginning Column Number}-{Ending Column Number}” (Example: “1-5, 8, 10-12”). If you want to import all of the columns in the file, enter two double quotes ("") here.

{data types} a list of special import data types with syntax “{Column Number}={Data Type}”, separated by commas (Example: “1=Text, 3=DateTime, 5=Text”). The column numbers entered correspond to the data types in the original data file, not the column numbers in the resulting **NCSS** dataset (although they will match unless you are importing from a file with multiple sheets or importing a data subset). The three possible data types for a column are *General*, *Text*, and *DateTime*. See the NCSS Data chapter in the documentation for more information about these data types. If a column will be imported with the default “General” data type, you do not need to include it in the list; only list columns that are to be imported with either *Text* or *DateTime* data types. If you want to import all columns with the same default “General” data type, enter two double quotes ("") here.

{p1} {p2} {p3} are extra parameters that are used for some file types, as described below.

Here is an example of importing an **NCSS** .s0 spreadsheet (The first column will be set to the *Text* data type):

Example

```
Import "NCSS"  "C:\Users\User\Documents\NCSS\abc.s0"  ""  "1=Text"
```

Here is an example of importing an **NCSS** .s0z database:

Example

```
Import "NCSS"  "C:\Users\User\Documents\NCSS\abc.s0z"  ""  ""
```

The following sections describe additional parameters that are required by some file types.

Access, SAS Transport

{p1} contains the index number or name of the table in the Access database that is to be imported. If this field is left blank, the first table will be imported. Only one table can be imported.

{p2} *SAS Transport only* - contains the source file encoding (e.g., “CP-1252”, “ISO-8859-15”, etc.). If this field is left blank, the default system file encoding will be used.

Here is an example of importing columns 1 through 10 of the third table from an Access database (The first column will be set to the *Text* data type and the third column will be set to the *DateTime* data type):

Example

```
Import "Access"  "C:\Users\User\Documents\NCSS\abc.mdb"  "1-10"  "1=Text,3=DateTime" "3"
```

Excel XLSX, Excel XLS, Lotus 123, Quattro

{p1} contains a list (separated by commas) of the index numbers or names of the sheets that are to be imported. If this field is left blank, only the first sheet will be imported.

{p2} *Lotus 123, Quattro only* - contains the source file encoding (e.g., “CP-1252”, “ISO-8859-15”, etc.). If this field is left blank, the default system file encoding will be used.

Here is an example of importing a sheet named “Sheet2” from an Excel spreadsheet (The third column will be set to the *Text* data type):

Example

```
Import "Excel"  "C:\Users\User\Documents\NCSS\abc.xlsx"  ""  "3=Text" "Sheet2"
```

dBase, Epi Info, Gauss, JMP, LimDep, Matlab, Minitab, Paradox, R, SAS, SPlus, SPSS, Stata, Statistica, Symphony, Systat

{p1} contains the source file encoding (e.g., “CP-1252”, “ISO-8859-15”, etc.). If this field is left blank, the default system file encoding will be used.

Here is an example of importing an SPSS file that uses the ISO-8859-15 encoding:

Example

```
Import "SPSS"  "C:\Users\User\Documents\NCSS\abc.sav"  ""  "" "ISO-8859-15"
```

ASCII Delimited

{p1} gives the delimiter that is used to separate values. For a blank space, enter "*blank*". For a tab character, enter "*Tab*" or "*TB*". For all other characters, enter the actual character (Examples: ";" or ","). Common choices are *blank*, *comma*, and *tab.*

{p2} gives the *Name Row* number. This is the number of the row in the file that contains the column names. If left blank, no column names are imported.

{p3} gives the *Lines per Record* number. This is the number of rows in the file that are devoted to a single row of data. Usually, this value is one. If left blank, this value is set to one.

Here is an example of importing a tab-delimited ASCII file with no column names:

Example

```
Import "Ascii_Delimited"  "C:\Users\User\Documents\NCSS\abc.txt"  ""  ""  "tab" 0 1
```

Here is an example of importing a comma-delimited ASCII file with column names on row 1:

Example

```
Import "Ascii_Delimited"  "C:\Users\User\Documents\NCSS\abc.txt"  ""  ""  "comma" 1 1
```

ASCII CSV

{p1} gives the *Name Row* number. This is the number of the row in the file that contains the column names. If left blank, no column names are imported.

{p2} gives the *Lines per Record* number. This is the number of rows in the file that are devoted to a single row of data. Usually, this value is one. If left blank, this value is set to one.

Here is an example of importing a CSV file with no column names:

Example

```
Import "Ascii_CSV"  "C:\Users\User\Documents\NCSS\abc.csv"  ""  ""  0 1
```

ASCII Fixed

{p1} contains the *Format* statement. This statement tells how a row of data from the file is to be interpreted. Details of how to create a format statement are contained in the topic on *Importing Fixed Format ASCII Files* in the Help system.

{p2} gives the *Name Row* number. This is the number of the row in the file that contains the column names. If left blank, no column names are imported.

{p3} gives the *Lines per Record* number. This is the number of rows in the file that are devoted to a single row of data. Usually, this value is one. If left blank, this value is set to one.

Here is an example of importing a fixed ASCII file with column names on row 2:

Example

```
Import "Ascii_Fixed"  "C:\Users\User\Documents\NCSS\abc.txt"  ""  ""  "2C4"  2  1
```

Export

This command exports an **NCSS** dataset to a file of a specified format. Exporting data is discussed in detail in the Exporting Data chapter. We refer you to that chapter for the details of data exporting.

The syntax of this command is

Export {type} {file name} {row save type} {column save type} {p1} {p2} {p3} {p4} {p5}

where

{type} a text variable or text phrase enclosed in double quotes that gives the type of data file to export. Some example file types are NCSS_S0, NCSS_S0z, Access, Excel_XLS, Excel_XLSX, ASCII_Delimited, and ASCII_CSV. See the Exporting Data chapter for a complete list of exportable file types.

{file name} a text variable or text phrase enclosed in double quotes that gives the name of the file into which the data are to be exported. Make sure the directory that will contain the exported file actually exists or macro execution will be interrupted.

{row save type} a variable indicating how the rows are to be exported (0 = export all rows, 1 = only export rows that pass the filter (ignored if no filter active), 2 = export only the user-specified rows). This parameter is optional for all output file types except those that require additional parameters p1, p2, etc. (if you wish to export a subset of columns but all rows, then you should enter a 0 here). If left out, all rows will be exported. When this option is set to 2, the proper syntax to specify the rows to export is "2;*<row list and/or range>*" (including the quotes) (Example: "2;1-25,30,50-70" would export rows 1 through 25, row 30, and rows 50 through 70).

{column save type}	a variable indicating how the columns are to be exported (0 = export all columns, 2 = export only the user-specified columns). This parameter is optional for all output file types except those that require additional parameters p1, p2, etc. If left out, all columns will be exported. When this option is set to 2, the proper syntax to specify the columns to export is "2;<*column list and/or range*>" (including the quotes). Column names or numbers can be entered. (Example: "2;C1-C3,C5,C6-C9" would export columns C1 through C3, column C5, and columns C6-C9. You could get the same result by entering "2;1-3,5,6-9").
{p1} {p2} {p3} {p4} {p5}	extra parameters that are used for some file types, as described next.

Here is an example of exporting the entire dataset to an Excel spreadsheet:

Example
```
Export "Excel"  "C:\Users\User\Documents\NCSS\abc.xlsx"
```

Here is an example of exporting a filtered row subset and all columns to an **NCSS** .s0 spreadsheet file:

Example
```
Export "NCSS"  "C:\Users\User\Documents\NCSS\abc.s0" 1
```

The following sections describe additional parameters that are required by some file types.

Access

{p1} gives the value of the *Maximum Text Width* parameter.

{p2} gives the value of the *New Table Base Name* parameter. To use the automatic setting, enter "".

Here is an example of exporting rows 1 through 100 and columns 1-5, 10, and 12-15 to an Access database file (The Access database table name will be "FromNCSS"):

Example
```
Export "Access"  "C:\Users\User\Documents\NCSS\abc.mdb" "2;1-100" "2;1-5,10,12-15" 20 "FromNCSS"
```

NCSS S0z

{p1} gives the value of the *Number of Empty "Add-On" Columns* parameter.

{p2} gives the value of the *Text Field Length* parameter.

{p3} gives the value of the *Number of Extra Text Fields* parameter.

{p4} gives the value of the *Extra Text Field Length* parameter.

Here is an example of exporting everything to an **NCSS** .s0z database file:

Example

```
Export "NCSS"  "C:\Users\User\Documents\NCSS\abc.s0z" 0 0 10 10 5 25
```

ASCII Delimited

{p1} gives the delimiter that is used to separate values. For a blank space, enter "*blank*". For a tab character, enter "*Tab*" or "*TB*". For all other characters, enter the actual character (Examples: ";" or ","). Common choices are *blank*, *comma*, and *tab.*

{p2} gives the *Export Column Names* option. This value is "1" for yes or "0" for no. If left blank, column names will not be exported.

{p3} gives the *Replace Missing Values With* option. For a blank space, enter "*blank*". To not replace missing values with anything, enter "". For all other characters, enter the actual character or phrase (Examples: "." or "Missing"). Common choices are "", *blank*, and *NA*.

{p4} gives the *Enclose Text Values Between* option. For double quotes, enter "DQ". For single quotes, enter "SQ". For all other characters, enter the actual character (Examples: "", " ' ", or "DQ"). To not enclose text values with anything, enter "".

{p5} gives the *Number of Characters per Line* option. Enter 0 for the number of characters to be chosen automatically by the program.

Here is an example of exporting all rows and only columns 1 through 10 to a comma-delimited ASCII file with column names, the word "NA" replacing missing values, and text values enclosed between double quotes:

Example

```
Export "Ascii_Delimited"  "C:\Users\User\Documents\NCSS\abc.txt" 0 "2;1-10" "," 1 "NA" "DQ" 0
```

ASCII CSV

{p1} gives the *Export Column Names* option. This value is "1" for yes or "0" for no. If left blank, column names will not be exported.

{p2} gives the *Replace Missing Values With*. For a blank space, enter "*blank*". To not replace missing values with anything, enter "". For all other characters, enter the actual character or phrase (Examples: "." or "Missing"). Common choices are "", *blank*, and *NA*.

{p3} gives the *Enclose Text Values Between* option. For double quotes, enter "DQ". For single quotes, enter "SQ". For all other characters, enter the actual character (Examples: "", " ' ", or "DQ"). To not enclose text values with anything, enter "".

{p4} gives the *Number of Characters per Line* option.

Here is an example of exporting everything to a comma-delimited CSV file with column names:

Example

```
Export "Ascii_CSV"  "C:\Users\User\Documents\NCSS\abc.csv" 0 0 1 "" "" 0
```

ASCII Fixed

{p1} contains the Format statement. This statement tells how a row of data from the file is to be interpreted. Details of how to create a format statement are contained in the topic on Importing Fixed Format ASCII Files in the Help system.

{p2} gives the *Export Column Names* option. This value is "1" for yes or "0" for no. If left blank, column names will not be exported.

{p3} gives the *Replace Missing Values With*. For a blank space, enter "*blank*". To not replace missing values with anything, enter "". For all other characters, enter the actual character or phrase (Examples: "." or "Missing"). Common choices are "", *blank*, and *NA*.

{p4} gives the *Number of Characters per Line* option.

Here is an example of exporting a filtered row subset and all columns to a fixed ASCII file with no column names:

Example

```
Export "Ascii_Fixed"  "C:\Users\User\Documents\NCSS\abc.txt" 1 0 "R8.2,X4,L4 /R10" 0 "" 0
```

ResizeRowsCols

This command resizes all rows and columns in the dataset according to the user-specified resize type. All three types create columns (rows) no narrower (shorter) than the default width (height). The resulting row heights and column widths are saved when the dataset is saved.

The syntax of this command is

ResizeRowsCols {rtype}

where

{rtype} is an integer corresponding to the type of row/column resizing to be done (0 = resize using defaults, 1 = resize using data and titles, 2 = resize using data only).

Following is an example of this command:

Example
ResizeRowsCols 1

SortBy {c1} {o1} {c2} {o2} {c3} {o3}

This command executes the Sort command on the currently open dataset. The dataset can be sorted by up to three columns in sequence. The entire dataset will be sorted, not just the selected columns.

The syntax of this command is

SortBy {c1} {o1} {c2} {o2} {c3} {o3}

where

{c1} is the name or number of the first column to sort by.

{o1} is the sort type (1 = ascending, 0 = descending).

{c2} is the name or number of the second column to sort by. This parameter is optional.

{o2} is the sort type (1 = ascending, 0 = descending). This parameter is only used of {c2} is used.

{c3} is the name or number of the third column to sort by. This parameter is optional.

{o3} is the sort type (1 = ascending, 0 = descending). This parameter is only used of {c3} is used.

The following are examples of this command:

Examples
SortBy C1 1
SortBy 1 1
SortBy "HeartRate" 1 "BP" 0

GetCell

This command obtains the value of a cell in the dataset. The syntax of this command is

GetCell {column} {row} {v}

where

{column} is the name or number of the column with the cell to be read.

{row} is the row number of the cell to be read.

{v} is the text or numeric macro variable that holds the value of the data cell.

The following are examples of this command:

Examples
GetCell "HeartRate" 27 H#
GetCell Name 16 Name16$

SetCell

This command sets a cell in the dataset to a specified value. The syntax of this command is

SetCell {column} {row 1} {row 2} {value}

where

{column} is the name or number of the column to receive the new value.

{row 1} is the first row in a range of rows to receive the new value.

{row 2} is the last row in a range of rows to receive the new value

{value} is the new value. This value may be text or numeric.

The following are examples of this command:

Examples
SetCell "HeartRate" 10 10 "100"
SetCell 1 10 20 100

NumRows

This command loads the number of rows of data in a particular column into a macro variable. The syntax of this command is

NumRows {column} {n}

where

{column} is a column name or number in the current dataset.

{n} is a numeric macro variable.

The following are examples of this command:

Examples
NumRows "HeartRate" n1#
NumRows 1 n#

NumCols

This command causes the number of columns contained in a list of dataset columns to be loaded into a macro variable.

The syntax of this command is

NumCols {col list} {n}

where

{col list} is an expression containing column names.

{n} is a numeric macro variable.

Following is an example of this command:

Example
NumCols "C2:C10, C15" ncols#

ColFromList

This command causes number of the i^{th} column in a list to be loaded into a macro variable.

The syntax of this command is

ColFromList {col list} {i} {n}

where

{col list} is an expression containing column names.

{i} is the item number to be selected from the column list.

{n} is a numeric macro variable that receives the number of the i^{th} item in the list.

The following are examples of this command:

Examples

```
ColFromList "C1,C2,C3,C10,C20" 4 V1#      (V1 becomes 10)
ColFromList "C1,C2,C3,C10,C20" 5 V2#      (V2 becomes 20)
```

GetMaxRows

This command loads the maximum number of rows with data in any column into a program variable.

The syntax of this command is

GetMaxRows {n}

where

{n} is a numeric macro variable.

The following are examples of this command:

Examples

```
GetMaxRows n1#
GetMaxRows n#
```

GetMaxCols

This command causes the number of the right-most column with data to be loaded into a macro variable.

The syntax of this command is

GetMaxCols {n}

where

{n} is a numeric macro variable.

The following are examples of this command:

Examples
GetMaxCols n1#
GetMaxCols n#

ColName

This command sets the name of the specified column.

The syntax of this command is

ColName {column} {name}

where

{column} is the current name or number of the column to be renamed.

{name} is the new name of the column. This name must follow standard **NCSS** column name restrictions.

The following are examples of this command:

Examples
ColName C1 "HeartRate"
ColName 1 "HeartRate"

ColLabel

This command sets the label of the specified column.

The syntax of this command is

ColLabel {column} {label}

where

{column} is the name or number of the column to be labeled.

{label} is the new label of the column.

The following are examples of this command:

Examples
ColLabel C1 "Heart Rate"
ColLabel 1 "Heart Rate"

ColDataType

This command sets the data type of the specified column.

The syntax of this command is

ColDataType {column} {data type}

where

{column} is the name or number of the column.

{data type} is the data type. The three possible data types for a column are *General*, *Text*, and *DateTime*.

The following are examples of this command:

Examples
ColDataType C1 "Text"
ColDataType 1 "DateTime"

ColFormat

This command sets the display format of the specified column.

The syntax of this command is

ColFormat {column} {format}

where

{column} is the name or number of the column to be formatted.

{format} is the format. See the Data Window chapter for more information about format statements.

The following are examples of this command:

Examples
ColFormat C1 "0.00"
ColFormat 2 "MM/dd/yyyy"

ColValueLabel

This command sets the value labels of the specified column.

The syntax of this command is

ColValueLabel {column} {vlab list}

where

{column} is the name or number of the column with the values to label.

{vlab list} is a list of value labels with syntax "{Value1}={Label1}", separated by commas. Value labels must be put in single quotes. The single quotes will be translated to double quotes when the macro is run. If you are using a computer language setting other than English that uses a comma as the decimal symbol, then the list should be separated by the computer's specified list separator (this is usually a semicolon if not a comma).

The following are examples of this command:

Examples

ColValueLabel C1 "0='No', 1='Yes'"	(English Language Setting)
ColValueLabel C1 "0='No'; 1='Yes'"	(German, etc. Language Setting)
ColValueLabel 1 "1='Low Value', 2='High Value'"	(English Language Setting)
ColValueLabel 1 "1='Low Value'; 2='High Value'"	(German, etc. Language Setting)

ColValueOrder

This command sets the value order of the specified column. The value order affects the order in which values are displayed in output.

The syntax of this command is

ColValueOrder {column} {value list}

where

{column} is the name or number of the column with the values to label.

{value list} is a list of values in the desired output display order, separated by commas. If you are using a computer language setting other than English that uses a comma as the decimal symbol, then the list should be separated by the computers specified list separator (this is usually a semicolon if not a comma).

The following are examples of this command:

Examples

```
ColValueOrder C1 "Yes, No"                  (English Language Setting)
ColValueOrder C1 "Yes; No"                  (German, etc. Language Setting)
ColValueOrder 1 "Low Value, High Value"     (English Language Setting)
ColValueOrder 1 "Low Value; High Value"     (German, etc. Language Setting)
```

ColTrans

This command sets the transformation of the specified column.

The syntax of this command is

ColTrans {column} {transform}

where

{column} is the name or number of the column to be transformed.

{transform} is the transformation formula.

The following are examples of this command:

Examples

```
ColTrans C1 "Log(C1)"
ColTrans 2 "C2*C3"
```

ColNote

This command sets the note of the specified column.

The syntax of this command is

ColNote {column} {note}

where

{column} is the name or number of the column for the new note.

{note} is the note.

The following are examples of this command:

Examples
ColNote C1 "Important information about this column."
ColNote 2 "Check this column for missing values!"

ColGroupBy

This command sets the group by value in the specified column. Group By columns (also known as Break Variables) are used to quickly divide your data into groups without using a Filter. When running a procedure, a separate analysis is conducted for each group, and separate statistical reports and plots are generated for each group. Group By columns should contain categorical data.

The syntax of this command is

ColGroupBy {column} {group by}

where

{column} is the name or number of the column for the group by value.

{group by} is the group by value. Enter "Yes" or "1" to make the specified column a "group by" column. Enter an additional numeric value in parentheses after "Yes" to specify the order that the columns are applied (e.g., "Yes (2)").

The following are examples of this command:

Examples

ColGroupBy C1 "Yes"	(Sets column C1 as a group by column.)
ColGroupBy 2 "Yes (2)"	(Sets column 2 as a group by column, applied second in order.)
ColGroupBy C1 ""	(Sets column C1 to no longer be a group by column.)

SetGroupByCols

This command sets the columns to group by and activates the group by system. Group By columns (also known as Break Variables) are used to quickly divide your data into groups without using a Filter. When running a procedure, a separate analysis is conducted for each group, and separate statistical reports and plots are generated for each group. Group By columns should contain categorical data.

The syntax of this command is

SetGroupByCols {col list}

where

{col list} is the list of columns to set at group by columns. The columns are applied on the order listed.

The following are examples of this command:

Examples
SetGroupByCols "C1"
SetGroupByCols "C1 C2"
SetGroupByCols "1 2"

SetGroupByActive

This command activates or deactivates the group by system.

The syntax of this command is

SetGroupByActive {yes no}

where

{yes no} is a number indicating whether or not the group by system is to be active: 0 = Inactive, 1 = Active.

The following are examples of this command:

Examples

SetGroupByActive 0	(Deactivates the Group By System)
SetGroupByActive 1	(Activates the Group By System)

SetFilter

This command sets a data filter and activates the filter system.

The syntax of this command is

SetFilter {filter}

where

{filter} is the filter statement to apply. The filter statement may include more than one column. See the Filters chapter for specific information about writing filters and appropriate syntax.

The following are examples of this command:

Examples

```
SetFilter "C1=1,2,3"
SetFilter "(C2>5) and (C2<=10)"
SetFilter "C4>C2+C3"
SetFilter "(C5>3) or (C6='Yes')"
```

SetFilterActive

This command activates or deactivates the current data filter.

The syntax of this command is

SetFilterActive {yes no}

where

{yes no} is a number indicating whether or not the filter is to be active: 0 = Inactive, 1 = Active.

The following are examples of this command:

Examples

```
SetFilterActive 0      (Deactivates the Filter System)
SetFilterActive 1      (Activates the Filter System)
```

SetFilterHideRows

This command shows or hides rows in the dataset that are excluded by the filter.

The syntax of this command is

SetFilterHideRows {yes no}

where

{yes no} is a number indicating whether or not filtered rows are to be hidden: 0 = show, 1 = hide.

The following are examples of this command:

Examples

```
SetFilterHideRows 0      (Show Filtered Rows)
SetFilterHideRows 1      (Hide Filtered Rows)
```

SetTransAutoRecalc

This command activates or deactivates the auto recalculation of data transformations.

The syntax of this command is

SetTransAutoRecalc {yes no}

where

{yes no} is a number indicating whether or not the filter is to be active: 0 = Inactive, 1 = Active.

The following are examples of this command:

Examples

```
SetTransAutoRecalc 0      (Turns Transformation Auto Recalculation OFF)
SetTransAutoRecalc 1      (Turns Transformation Auto Recalculation ON)
```

RunTrans

This command executes all transformation in the current dataset. This is only necessary if *Auto Recalculate Transformations* is turned off.

The syntax of this command is

RunTrans

Following is an example of this command:

Example

```
RunTrans
```

Procedure Commands

The following commands open, modify, run and close procedures.

List of Procedure Commands:

LoadProc
UnloadProc
RunProc
LoadProcSettings
SaveProcSettings
NewProcSettings
Option

LoadProc

This command loads the designated procedure window. Once loaded, the options of the procedure may be modified and then the procedure can be executed.

The syntax of this command is

LoadProc {proc}

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom left of each procedure window when *Procedure Info* is selected to show. To display procedure info, click *View* > *Procedure Info* from the procedure window menu.

The following are some examples of valid *LoadProc* commands:

Examples
LoadProc 24
LoadProc DescStat

UnloadProc

This command closes the indicated procedure window. The syntax of this command is

UnloadProc {proc}

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom left of each procedure window when *Procedure Info* is selected to show. To display procedure info, click *View* > *Procedure Info* from the procedure window menu.

The following are some examples of valid *UnloadProc* commands:

Examples
UnloadProc 24
UnloadProc DescStat

RunProc

This command executes the indicated procedure. The syntax of this command is

RunProc {proc}

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom left of each procedure window when *Procedure Info* is selected to show. To display procedure info, click *View* > *Procedure Info* from the procedure window menu.

The following are some examples of valid *RunProc* commands:

Examples
RunProc 24
RunProc DescStat

LoadProcSettings

This command loads the settings from a procedure settings file. Once loaded, the options of the procedure may be modified and then the procedure can be executed.

The syntax of this command is

LoadProcSettings {proc} {file name}

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom left of each procedure window when *Procedure Info* is selected to show. To display procedure info, click *View* > *Procedure Info* from the procedure window menu.

{file name} is a required variable or text constant that gives the name of the settings file. This item should at least contain the settings file name and extension (Example: "ProcSettings.t24"). If only the settings file name is given, then **NCSS** will look for the settings file in the same folder where the macro resides. If the complete path and file name is given (Example: "C:\Users\User\Documents\ProcSettings.t24"), then **NCSS** will load the settings file from anywhere on the computer or network.

This function replaces the *LoadTemplate* function, which will also work with the same arguments.

The following are some examples of valid *LoadProcSettings* commands:

Examples

```
LoadProcSettings DescStat "ProcSettings1.t24"     (This will load the settings file from the folder with the
                                                   macro)
LoadProcSettings DescStat "C:\Users\User\Documents\ProcSettings1.t24"
```

SaveProcSettings

This command saves the settings of the specified procedure to a settings file.

The syntax of this command is

SaveProcSettings {proc} {file name}

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom left of each procedure window when *Procedure Info* is selected to show. To display procedure info, click *View* > *Procedure Info* from the procedure window menu.

{file name} is a required variable or text constant that gives the name of the settings file. This item should at least contain the settings file name and extension (Example: "ProcSettings.t24"). If only the settings file name is given, then **NCSS** will save the settings file in the same folder where the macro resides. If the complete path and file name is given (Example: "C:\Users\User\Documents\ProcSettings.t24"), then **NCSS** will save the settings file anywhere on the computer or network.

This function replaces the *SaveTemplate* function, which will also work with the same arguments.

The following are some examples of valid *SaveProcSettings* commands:

Examples

```
SaveProcSettings DescStat "ProcSettings1.t24"      (This will save the settings file into the folder with the
                                                    macro)
SaveProcSettings DescStat "C:\Users\User\Documents\ProcSettings1.t24"
```

NewProcSettings

This command resets the settings of the specified procedure to program defaults.

The syntax of this command is

NewProcSettings {proc}

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom left of each procedure window when *Procedure Info* is selected to show. To display procedure info, click *View* > *Procedure Info* from the procedure window menu.

This function replaces the *NewTemplate* function, which will also work with the same arguments.

Here is an example of a valid *NewProcSettings* command:

Example
NewProcSettings DescStat

Option

This command lets you set the values of the individual options of a procedure. For example, you may want to change the name of the variable that is to be processed.

The syntax of this command is

Option {proc} {number} {value 1} {value 2} {value 3} ...

where

{proc} is a variable or constant that gives the name or number of the procedure. Each procedure's name and number is displayed near the bottom left of each procedure window when *Procedure Info* is selected to show. To display procedure info, click *View* > *Procedure Info* from the procedure window menu.

{number} is the number of the option that is to be set. The option numbers for each item are displayed near the bottom left of each procedure window when *Procedure Info* is selected to show. To display procedure info, click *View* > *Procedure Info* from the procedure window menu.

{value 1} is the new value of the option.

{value 2}... are the new values of the remaining parameters of the option. Most options only have one value, so a second value is not necessary. However, a few options, such as text properties, bring up a window for option selection. These options have two or more parameters.

The following are some examples of valid OPTION commands:

Examples
Option 24 2 4
Option DescStat 4 "HeartRate"

Output Commands

The following commands manage the output windows.

List of Output Commands:

SaveOutput
ClearOutput
PrintOutput
AddToGallery
OpenGallery
SaveGallery
ClearGallery
PrintGallery

SaveOutput

This command saves the current output to the designated file name.

The syntax of this command is

SaveOutput {file name}

where

{file name} a text constant or variable that gives the name of the file to receive the output. Note that the extension of the file name should be ".RTF".

Following is an example of this command:

Example
SaveOutput "C:\Users\User\Documents\Sample.rtf"

ClearOutput

This command clears (erases) the current output window.

The syntax of this command is

ClearOutput

Following is an example of this command:

Example
ClearOutput

PrintOutput

This command prints the current output.

The syntax of this command is

PrintOutput {with dialog}

where

{with dialog} an optional numeric constant indicating whether or not to print with a dialog (0 = print without dialog, 1 = print with dialog). If left blank, this command will print one copy of all pages of the output to the default printer. If you print with a dialog, the macro will be interrupted for the user to complete the dialog window.

The following are examples of this command:

Examples

PrintOutput	(Prints without a dialog)
PrintOutput 1	(Prints with a dialog)

AddToGallery

This command copies the output in the output window to the gallery window. Note that nothing is saved by this command.

The syntax of this command is

AddToGallery

Following is an example of this command:

Example

AddtoGallery

OpenGallery

This command opens and displays the contents of the specified file.

The syntax of this command is

OpenGallery {file name}

where

{file name} a text constant or variable that gives the name of the file to opened. Note that only RTF files can be opened.

Following is an example of this command:

Example
OpenGallery "C:\Users\User\Documents\NCSS\Sample.rtf"

SaveGallery

This command saves the current contents of the gallery window to the designated file name.

The syntax of this command is

SaveGallery {file name}

where

{file name} a text constant or variable that gives the name of the file to receive the log. Note that the extension of the file name should be ".RTF".

Following is an example of this command:

Example
SaveGallery "C:\Users\User\Documents\NCSS\Reports\Sample.rtf"

ClearGallery

This command clears the gallery window.

The syntax of this command is

ClearGallery

Following is an example of this command:

Example
NewGallery

PrintGallery

This command prints the current gallery.

The syntax of this command is

PrintGallery {with dialog}

where

{with dialog} an optional numeric constant indicating whether or not to print with a dialog (0 = print without dialog, 1 = print with dialog). If left blank, this command will print one copy of all pages of the gallery to the default printer. If you print with a dialog, the macro will be interrupted for the user to complete the dialog window.

The following are examples of this command:

Examples

PrintGallery	(Prints without a dialog)
PrintGallery 1	(Prints with a dialog)

Alphabetical Macro Command List

AddToGallery
ClearGallery
ClearOutput
ColDataType {column} {data type}
ColFormat {column} {format}
ColFromList {col list} {i} {n}
ColGroupBy {column} {group by}
ColLabel {column} {label}
ColName {column} {name}
ColNote {column} {note}
ColTrans {column} {transform}
ColValueLabel {column} {vlablist}
ColValueOrder {column} {valuelist}
DataNew
DataOpen {file name}
DataSave {file name} {withfilter}
End
Export {type} {file name} {with filter} {p1} {p2} {p3} {p4} {p5}
{flag}:
GetCell {column} {row} {v}
GetMaxCols {n}
GetMaxRows {n}
GoTo {p1}
Heading {h1}
If {p1} {logic} {p2} GoTo {p3}

Import {type} {file name} {columns} {data types} {p1} {p2} {p3}
Input {variable} {prompt} {title} {default}
LoadProc {proc}
LoadProcSettings {proc} {file name}
NewProcSettings {proc}
NumCols {col list} {n}
NumRows {column} {n}
OpenGallery {file name}
Option {proc} {number} {value 1} {value 2} {value 3} ...
Print {p1} {p2} {p3} ...
PrintGallery {with dialog}
PrintOutput {with dialog}
ResizeRowsCols {rtype}
RunProc {proc}
RunTrans
SaveGallery {file name}
SaveOutput {file name}
SaveProcSettings {proc} {file name}
SendKeys {value}
SetCell {column} {row 1} {row 2} {value}
SetFilter {filter}
SetFilterActive {yes no}
SetFilterHideRows {yes no}
SetGroupByActive {yes no}
SetGroupByCols {col list}
SetTransAutoRecalc {yesno}
SortBy {c1} {o1} {c2} {o2} {c3} {o3}
UnloadProc {proc}
WindowLeft {value}
WindowTop {value}

Examples

The following section provides examples of **NCSS** macros. Our intention is that these examples will help you learn how to write macros to accomplish various repetitive tasks with **NCSS**. In all examples, “%p%” in the DataOpen command is replaced by the folder that was used for installation.

Example 1 – Automatically Run a Procedure

This macro opens a dataset and calculates descriptive statistics on the first four columns of that dataset.

```
'***  Open the Fisher dataset.
DataOpen "%p%\Example Data\Fisher.NCSS"

'***  Load the Descriptive Statistics procedure. (Note that the default values are used.)
LoadProc DescStat

'***  Specify that the first four columns are to be analyzed.
Option DescStat  1 "1:4"

'***  Run the analysis.
RunProc DescStat
```

Example 2 – Run Several Procedures

This macro opens a dataset, calculates descriptive statistics on the first two columns, and then runs a linear regression using those two columns.

```
'***  Open the Fisher dataset.
DataOpen "%p%\Example Data\Fisher.NCSS"

'***  Load the Descriptive Statistics procedure. (Note that the default values are used.)
LoadProc DescStat

'***  Specify that the first two columns are to be analyzed.
Option DescStat  1 "1:2"

'***  Run the analysis.
RunProc DescStat

'***  Load the Linear Regression procedure. (Note that the default values are used.)
LoadProc LinReg

'***  Specify SepalLength as the Y Variable (Option 1) and SepalWidth as the X Variable (Option 2)
Option LinReg 1 SepalLength
Option LinReg 2 SepalWidth

'***  Run the analysis.
RunProc LinReg
```

Example 3 – Simple Looping

This macro opens a dataset and calculates descriptive statistics on the first four columns of that dataset. Unlike Example 1, this macro uses a simple loop to step through the four columns.

Notice that the loop finishes once the variable I# is equal to 4.

```
'***  Open the Fisher dataset.
DataOpen "%p%\Example Data\Fisher.NCSS"

'***  Initialize the counter variable.
I#=0

'***  Loop through the four columns.
Flag1:

    '***  Increment the counter variable.
    I#=I#+1

    '***  Load the Descriptive Statistics procedure. (Note that the default values are used.)
    LoadProc DescStat

    '***  Specify that the Ith column is to be analyzed.
    Option DescStat 1 I#

    '***  Run the analysis.
    RunProc DescStat

If I#<4 GoTo Flag1
```

Example 4 – Using NumCols and ColFromList

This macro opens a dataset and calculates descriptive statistics on three non-contiguous columns. The NumCols command is used to load a numeric variable n# with the number of columns.

```
'***  Open the Fisher dataset.
DataOpen "%p%\Example Data\Fisher.NCSS"

'***  Load the Descriptive Statistics procedure. (Note that the default values are used.)
LoadProc DescStat

'*** Turn off all but the Summary Section report.
Option DescStat 16 1
Option DescStat 17 0
Option DescStat 18 0
Option DescStat 19 0
Option DescStat 20 0
Option DescStat 42 0
Option DescStat 43 0
Option DescStat 44 0
Option DescStat 45 0
Option DescStat 46 0
Option DescStat 47 0
Option DescStat 48 0
Option DescStat 49 0

'*** Load the column list into V$.
V$="SepalLength   PetalLength   PetalWidth"

'*** Load the number of columns into n#.
NumCols V$ n#

'***  Initialize the counter variable.
I#=0

'***  Loop through the columns.
Flag1:
    '***  Increment the counter variable.
    I#=I#+1

    '*** Load the Ith column from the list.
    ColFromList V$ I# name$

    '*** Set the Column to name$.
    Option DescStat 1 name$

    '***  Run the analysis.
    RunProc DescStat

If I#<n# GoTo Flag1
```

Example 5 – Multiple Analyses using a Group By Column

This macro opens a dataset and runs separate simple linear regression analyses and scatter plots for three groups within the dataset. The three groups are defined by the “Iris” column in the dataset, which is set as a “Group By” column. Thus, the macro does not require knowledge of the group names, nor does it require knowledge of the number of groups to be analyzed.

```
'***  Open the Fisher dataset.
DataOpen "%p%\Example Data\Fisher.NCSS"

'***  Set column Iris as a group by column.
SetGroupByCols "Iris"

'***  Run the Linear Regression procedure.
LoadProc LinReg
RunProc LinReg

'***  Run the Scatter Plot procedure.
LoadProc ScatPlot
RunProc ScatPlot

'***  Deactivate the group by system.
SetGroupByActive 0

'***  End of Macro.
Print ""
Print "MACRO COMPLETE"
```

Example 6 – Multiple Analyses using Filters

This macro opens a dataset and runs separate simple linear regression analyses and scatter plots for three groups within the dataset. The three groups are defined by the “Iris” column in the dataset. The macro finds the levels of the “Iris” column and uses the resulting values to determine the groups. Thus, the macro does not require knowledge of the group names, nor does it require knowledge of the number of groups to be analyzed.

```
'***  Open the Fisher dataset.
DataOpen "%p%\Example Data\Fisher.NCSS"

'***  Obtain the unique values of the Iris column and put them in C6.
ColTrans C6 "Uniques(Iris)"
RunTrans

'***  Obtain the number of unique Iris values using C6.
NumRows C6 n#

'***  Activate the filter system.
SetFilterActive 1

'***  Initialize the counter variable.
I#=0

'***  Loop through the unique Iris values for grouping.
Flag1:
    '***  Increment the counter variable.
    I#=I#+1

    '***  Get the Ith value from C6.
    GetCell C6 I# IrisGroup$

    '***  Filter the values according to the Iris column.
    Filter$ = "Iris = " & IrisGroup$
    SetFilter Filter$

    '***  Run the Linear Regression procedure for the filtered group.
    LoadProc LinReg
    RunProc LinReg

    '***  Run the Scatter Plot procedure for the filtered group.
    LoadProc ScatPlot
    RunProc ScatPlot

If I#<n# GoTo Flag1

'***  Deactivate the filter system.
SetFilterActive 0

'***  End of Macro.
Print ""
Print "MACRO COMPLETE"
```