28 Creating a Simple Macro

To Table of Contents

To Index

What Is a Macro?

MINITAB is usually used interactively, which means that each command carries out as soon as you click OK in a dialog box or enter it in the Session window. You can also use a MINITAB macro—a set of MINITAB commands stored in a file—to automate a repetitive task, such as generating a monthly report, or to extend MINITAB's functionality, such as computing a special test statistic. In other words, you can write macros tailored to your needs.

Once you create a macro file, you invoke it in the Session window or Command Line Editor. Type **%** followed by the macro file name, as in %mymacro. For details, see *Invoking a Macro* on page 28-5.

Terminology: three types of macros

If you look at other chapters in the macro section of this manual, or in on-line Help, you may see terms that distinguish between the three types of MINITAB macros:

- "**Global macros**" refers to the simplest form of macro, the type discussed in this chapter.
- "**Local macros**" refers to the more sophisticated form of macro, discussed in Chapter 29, *Advanced Macros*.
- "**%macros**" refers to both global and local macros. Because both types are invoked by typing **%**, end in the extension MAC, and share many capabilities, the two types are often discussed together.
- "**Execs**" are an older form of MINITAB macro, and are described in Chapter 33, *Using Execs*.

For details on the differences between the three types of macros, see *Advanced Macros Overview* on page 29-2.

The Structure of a Simple Macro

A macro consists of lines of text stored in a text file. The simplest form of macro, called a global macro, follows this structure:

GMACRO
template
body of the macro
ENDMACRO

GMACRO and ENDMACRO

These mark the beginning and end of each macro. GMACRO must be the first line of your macro. GMACRO says that this is a global macro, not a local macro. GMACRO and ENDMACRO cannot be abbreviated.

Template

The template is the name of your macro; you choose the name, such as MyMacro or Analyze. The template can contain letters, numbers, or the underscore character, but must start with a letter. The template can be upper, lower, or mixed case; MINITAB ignores case when you invoke the macro.

It is probably most convenient, but it is not required, to use the macro file name as your template. For example, all of the following are valid combinations of templates and file names.

Template	File name	Invoked by
MyMacro	MYMACRO.MAC	%MYMACRO
Analyze	TEST.MAC	%TEST
Analyze2	TEST2.TXT	%TEST2.TXT[a]

a. When invoking a macro, you only have to include the file extension if it is not the default of MAC. See *Invoking a Macro* on page 28-5.

Body of the macro

The body of the macro consists of MINITAB commands, macro statements (documented in Chapters 30 and 31), and invocations of other global macros (see *Invoking Macros from Within Macros* on page 30-7).

Example of a Simple Macro

Here is a simple example of a macro file named ANALYZE.MAC. Indenting is not necessary, but may be done to improve readability as illustrated here.

```
GMACRO
Analyze

NAME C1='Yield' C2='Chem1' &
     C3='Chem2' C5='Ln.Yield'
PRINT C1-C3
DESCRIBE C1-C3
LET C5 = LOGE('Yield')
REGRESS C5 2 C1 C2

ENDMACRO
```

Marks the beginning of the global macro.

The template, or the name, of this macro.

Body of the macro.

Marks the end of the macro.

Chapters 30 and 31 contain more examples of macros.

Tip **Use &, the continuation symbol**

If you want to continue a macro statement onto another line, end the first line with the symbol &, just as you do with commands in interactive MINITAB.

Creating a Macro

▶ To create a macro

1. Use any text editor or word processor to write your macro.
2. Save the macro file as a text file. Give the file any name you wish, and the file extension MAC (if you want to use a different file extension, see *Notes on saving macro files* below).
3. Save the macro file to the Macros subdirectory of your main MINITAB directory (if you want to use a different directory, see *Notes on saving macro files* below).

▶ To create a macro from a MINITAB session

1. Go through your MINITAB session, executing commands interactively, as usual.
2. Choose **Window ➤ History** to open MINITAB's History window. This window displays the most recent commands (just commands, not output) executed in your session.
3. Highlight the commands you want to include in your macro, and choose **Edit ➤ Copy**.
4. Open any word processing application and choose **Edit ➤ Paste**.

5 Change any commands if you wish. Then insert three lines:

GMACRO	at the top of the file
template	a name you choose as the second line in the file.
ENDMACRO	at the bottom of the file

6 Save the changes in text format with whatever file name you wish and the file extension MAC (if you want to use a different file extension, see *Notes on saving macro files* below).

7 Save the macro file to the Macros subdirectory of your main MINITAB directory (if you want to use a different directory, see *Notes on saving macro files* below).

Notes on saving macro files

- If you are using a word processor, make sure you do not save the file in that application's native format. For example, if you used Microsoft Word to write the macro, do not save the file as a Word document. Choose **File ➤ Save As** and select a file type of text only.
- When you name the file, you may use any file extension, but MAC is the default extension MINITAB looks for when you invoke the macro.
- You can save the file to any directory, but MINITAB looks automatically in the macros subdirectory of the main MINITAB directory. If you save the file in a different directory, you must specify the full path when you invoke the macro, as in %C:\WORK\MYMACRO.

Invoking a Macro

To invoke a global macro from MINITAB, enter the symbol % followed by the macro file name. For example, to invoke a macro file named ANALYZE.MAC, enter the command:

```
%ANALYZE
```

Notes on invoking macros

- The default file name extension for macros is MAC. When you invoke a macro that has an extension of MAC, you only need to type the file name, as in %ANALYZE. If the extension is not MAC, you must type the file name and extension, as in %ANALYZE.TXT.
- When you invoke a macro, by default MINITAB looks for that macro file first in the current directory, then in the macros subdirectory. If the macro is not in one of those

default directories, you can specify the directory by including a path when you invoke the macro. For example, `%C:\SALES\ANALYZE`.

- If a files name includes spaces, put the name in single quotes, as in `%'a very long file name.MAC'`

Here are some examples of the different ways to invoke a macro, with details on where MINITAB will look first for a macro file:

`%ANALYZE`	MINITAB looks for the file ANALYZE.MAC in your current directory first, then in the \MACROS subdirectory of your main MINITAB directory.
`%C:\SALES\ANALYZE`	MINITAB looks for the file ANALYZE.MAC in the \SALES subdirectory on the C drive.
`%TEST.CMD`	MINITAB looks for the file TEST.CMD (where CMD is an extension created by the user) in your current directory first, then in the \MACROS subdirectory of your main MINITAB directory.
`%'A Very Long Macro Name'`	MINITAB looks for the file A VERY LONG MACRO NAME.MAC in your current directory first, then in the \MACROS subdirectory of your main MINITAB directory.

Adding Comments

You can annotate your macro program by using the comment symbol # and the NOTE command.

To add comments that do not display in the Session window

Place the symbol # anywhere on a line to tell MINITAB to ignore the rest of the line. Text after # is not displayed in the Session window when the macro is executed (even when you use the ECHO mode described on page 32-4). For example:

```
# This is a comment on its own line
PRINT C1-C3 # This is a comment
```

To add comments that display in the Session window

Put the NOTE command at the beginning of a line. All text on that line will be ignored by the macro processor. However, text on a NOTE line (except the first five spaces—the word NOTE and a space) does display in the Session window when the

macro is executed. To display a blank line, include a line containing only the word NOTE.

▶ Example of using NOTE in a macro

Macro code

```
NOTE Here come the data
NOTE
PRINT C1-C3
```

Results in Session window

```
Here come the data

Data Display
 Row  Yield  Chem1  Chem2
   1  11.28     87   1.83
   2   8.44     61  25.42
   3  13.19     59  28.64
...
```

Tip **Increase readability by adding blank lines**

NOTE can add blank lines to make your output more readable. But you can also make your macro file more readable by adding blank lines between the lines of macro statements and commands. The blank lines will not interfere with the execution of the macro, and will not appear in the Session window. You do not have to start a blank line with a # symbol.

Adding Control Statements

Control statements can make your macro flexible and powerful. For example, if you want the macro to…

- perform some action only if some condition is true or false, use an IF statement
- perform some action a set number of times, use a DO–ENDDO loop
- repeat a block of commands as long as some condition is true, use a WHILE–ENDWHILE loop.
- start another macro from within your macro, use CALL and RETURN

These control statements and others are detailed in Chapter 30, *Controlling Macro Flow*.

▶ Example of using IF in a macro

You can make your macro conditionally execute commands with the IF control statements—IF, ELSEIF, ELSE, and ENDIF. The IF control statements can evaluate a logical expression using comparison operators (such as > for "greater than" and = for "equals") and Boolean operators (such as AND, OR, and NOT). For more details, see *IF, ELSEIF, ELSE, ENDIF* on page 30-2.

In the example below, the macro only executes the REGRESS command if there are more than three observations in the column Yield. If there are less than three or exactly three observations, the macro prints a message in the Session window using the NOTE command (described on page 28-6).

```
LET K1 = COUNT('Yield')
IF K1 < 3
NOTE Not enough observations
ELSEIF K1 = 3
NOTE Add one observation
ELSE
  REGRESS C5 2 C1 C2
ENDIF
```

Finding Problems in Macros

If your macro produces unexpected results or generates an error message, MINITAB provides several tools to help you track down and correct the problem.

Chapter 32, *Handling Errors in Macros*, details how MINITAB responds to errors, describes tools that can help you find these problems, and lists the MINITAB commands that act differently in macros than in interactive MINITAB.

Here are some common problems you may want to check for first:

- The syntax used in the macro is not correct—for example, the macro does not begin with GMACRO or end in ENDMACRO.
- The MINITAB commands in the macro are not correct—for example, the REGRESS command is misspelled, or a column name is provided when the command expected a constant. This kind of mistake generates the same kind of error message you would have received if you were using MINITAB in interactive mode.
- The macro uses a MINITAB command that works differently in a macro than in interactive MINITAB—see *MINITAB Commands that Work Differently in Macros* on page 32-7.

A Macro that Starts Automatically

You can create a special file called STARTUP.MAC which executes automatically every time you start or restart MINITAB. A startup macro is a handy tool if you wish to avoid typing the same commands every time you start a MINITAB session.

STARTUP.MAC can be a global macro (the type discussed in this chapter) or local macro (discussed in Chapter 29, *Advanced Macros*). Users of earlier versions of

MINITAB may have an Exec file named STARTUP.MTB which serves the same purpose and will still work (see Chapter 33, *Using Execs*).

When you start or restart MINITAB, MINITAB looks for macro files in the order shown below, and executes the first one it finds, if one exists.

1. STARTUP.MAC in your current directory
2. STARTUP.MTB in your current directory
3. STARTUP.MAC in the Macros subdirectory of the main MINITAB directory
4. STARTUP.MAC in the Macros subdirectory of the main MINITAB directory

Create a STARTUP.MAC file as you would any other macro, with a text editor or word processor. The file must be called STARTUP.MAC. It can be written as a global or a local macro.

Possible applications of the startup file include:

- Set up an OUTFILE or JOURNAL.
- Set switches (BRIEF, OH, OW).
- Set up graphics options (GPRO, GSTD).
- Automatically RETRIEVE a saved worksheet.
- Send users messages via the NOTE command.

What Next?

The simple macros discussed in this chapter demonstrate only a few of the things you can do with MINITAB macros. The other chapters in this part of *MINITAB User's Guide 1* describe the rest of MINITAB's macro capabilities:

- Chapter 29, *Advanced Macros*, shows you how to create more sophisticated macros that act just like regular MINITAB commands, with arguments and subcommands. These advanced macros can also create temporary data that do not have to be stored in your worksheet—for example, if your macro uses a list of random numbers, you do not have to first store the data in a column and then erase it by the end of the macro.
- Chapter 30, *Controlling Macro Flow*, describes techniques and commands you can use to control which commands are executed, and when.
- Chapter 31, *Managing Input and Output*, shows you how to make a macro interactive, label output, save data, and more.

- Chapter 32, *Handling Errors in Macros*, discusses how to interpret error messages, which MINITAB commands behave differently in macros, and tools you can use to track down and correct problems.
- Chapter 33, *Using Execs*, discusses MINITAB's older macro functionality.