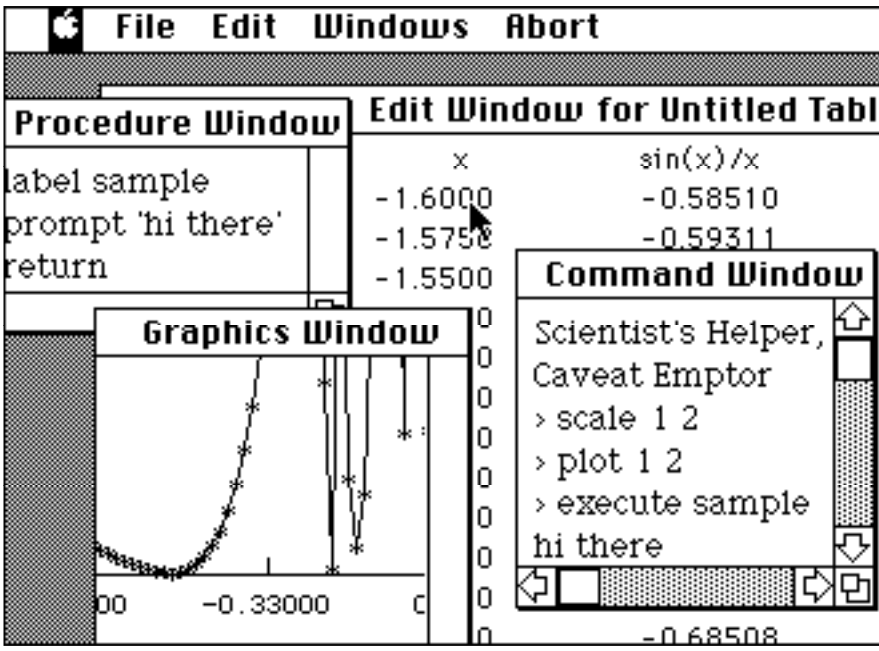
Scientist's Helper

User's Manual
MacIntosh Version 2.4
for the 512K MacIntosh and Mac XL

by William Menke
College of Oceanography
Oregon State University
Corvallis OR 97331
(503) 754-2912

January 4, 1986

A Data Manipulation program designed especially for
Scientists and Engineers
for performing
arithmetic, curve fitting and time series analysis
on tabular data



Introduction. Scientist's Helper is an interactive data manipulator designed for the kind of tabular data commonly used by scientists and engineers. The fundamental data structure in Scientist's Helper is a table of numbers. These data can be input, saved as files, viewed, plotted, and operated upon mathematically.

p Scientist's Helper is basically a command string oriented program. The command set includes graphics, arithmetic, time-series analysis, curve fitting and table management operations.

p Scientist's Helper makes full use of the MacIntosh windowing and menu selection functions. For instance, it contains a mouse-driven table editor to facilitate inputing, viewing, and manipulating the table.

p The contents of the table can be plotted on the Mac screen and then saved as a MacPaint format file.

p Procedures (programs) can be written in RegTab's command language and then executed as if they were Scientist's Helper commands. These procedures can use string variables and a variety of control structures.

The Scientist's Helper Table. A Scientist's Helper table can contain up to 4096 rows and 32 columns of single-precision floating-point data. In addition to the usual numbers, table entries can be set to NaN (for not a number). Improper mathematical operations such as division by zero will generate these entries. Thus, Scientist's Helper will not crash from an arithmetic error.

Scientist's Helper tables come in two variations. One type is 'interpolated', meaning that the data in column 1 increase linearly with row number. Entries in column 1 of interpolated tabes cannot be altered. The other type is uninter- polated, in which column 1 is no different than any other column. Interpolated tables require less disk storage than uninterpolated ones.

A header is associated with the table. The header contains the basic information about the table, and is stored along with the table when a file containing a table is created. The header contains the following information, which are referred to by standard names:

rows: the number of rows in the table;
cols: the number of columns in the table;
title: an 80 character string describing the data;
interpolated: a boolean flag indicating whether or not the table is interpolated;
colname: a 10 character name for each column;
samp, start: the sampling interval and starting value of the table if interpolated. Column 1 is computed by the rule start + samp–(row-1).

The amount of the computer's memory allocated for the table can be varied. The default allocation when Scientist's Helper is first run is 128 rows and 2 columns. This size can be increased by the **allocate** command, up to 4096 rows and 32 columns. This should be done at the beginning of the session, since reallocating the table destroys the data in the old one (unless it is first saved in a file). The amount of allocated memory controls the maximum permissible size of the table. The maximum size of a table should not be confused with its current size given by **rows** and **cols,** which may be less than or equal to the maximum size.

Windows. Scientist's Helper has four windows, named Command Window, Graphics Window, Edit Window, and Procedure Window:

The **Command Window** displays a history of what has been done during the session, and is where new commands are typed. The contents of the Command Window can be freely edited. Only text added to the last line and followed by a return are interpreted as commands.

The **Graphics Window** contains a plot of columns of the table. Plots are made by typing the appropriate commands in the command window. Clicking the mouse when the graphics window is active causes the current position of the cursor to be printed in the command window.

The **Edit Window** displays the table in tabular form and allows you to edit its entries.

The **Procedure Window** contains currently defined procedures (programs). The can be freely edited and transferred between the window and files.

Menu Items. In addition to the desk accessories, their are several menus:

Apple, through wich the desk accessories can be accessed.

File, through which all disk I/O is accomplished. Note that tables can be stored on disks in two forms: Binary and Ascii. You should normally use the Binary form, since it is faster and the entire table header is saved. The ascii form is useful for transferring data between Scientist's Helper and other programs. These files contain only the column names and table values, with the items on each line separated by tabs. Note that MacWrite files containing tables can be read into Scientist's Helper if they are saved in the text-only mode.

Edit, which contains the standard undo, copy, cut, paste, and clear commands. Note that the undo and clear commands do nothing - they are provided only for use of the desk accessories.

Windows, which provides a way to select hidden windows.

Abort, which allows procedures and some commands to be aborted.

Command Strings. Many Scientist's Helper operations are invoked by typing command strings in the Command Window. Scientist's Helper command strings contain up to six command words (where a word is a sequence of characters containing no blanks or a quoted sequence of characters containing blanks). Command words can be either predefined keywords or parameters.

A Typical Scientist's Helper Session. In the following tutorial, material Scientist's Helper types is printed in **bold**, material the user types is printed in plain text, and comments are printed in *italic*.

Scientist's Helper, Version 2.4, by William Menke
Caveat Emptor

New Table 128 by 2

>	allocate 128 3	*creates 128 by 3 table*
>	title 'my test dataset'	*sets title*
>	colname 1 'time, t'	*labels column 1*
>	colname 2 '0.987sin(t)'	*labels column 2*
>	colname 3 'sin(t)/t'	*labels column 3*
>	samp 0.1	*column 1 is interpolated column*
>	start 0.0	*with sampling interval 0.1*
>	interpolated true	*staring value 0.0*
>	cfunction sin 1 2	*put 0.987 times sine of column 1*
>	cmath 2 *# 0.987 = 2	*into column 2*
>	cmath sin 1 3	*put sine of column 1 divided by*
>	cmath 3 / 1 = 3	*column 1 into column 3*
>	table 1 3 1.0	*sin(0)/0 now set to NaN, reset to 1*
>	xaxis 0 12.8	*set abcissa of plotting screen*

>	yaxis -1 1	*set ordinate of plotting screen*
>	clear	*clear graph*
>	plot 1 2 solid	*plot column 2 against 1*
>	plot 1 3 dotted	*plot column 3 against 1*
>	quit	*quit from Scientist's Helper*

Scientist's Helper Variables. Variables contain strings of up to 80 characters in length. A variable's name can be any string that does not contain a blank. Once defined (eg. by the **setvar** command), the value of a variable can be used as a command word by including its name in the command string prefaced by the @ symbol. For example, the commands:

```
setvar   date   'November  1, 1755'
prompt @date
```

create a variable named 'date' that is set to the value 'November 1, 1755', and then types the value of the variable in the command window. Note that variables can contain numbers in string form. Variables are mainly useful in procedures. In addition to variables defined by the user, Scientist's Helper also defines and automatically updates variables set to commonly used parameters:

The header variables: **rows, cols, title, interpolated, samp, start;**

The arguments of the last execute procedure command: **arg1, arg2, arg3, arg4;**

The endpoints of the graphics axes, **xmin, xmax, ymin, ymax;**

The position of the cursor in user coordinated after the last cursor command: **xpos, ypos;**

The minimum value in a column after the min command: **min;**

The maximum value in a column after the max command: **max;**

The mean, standard deviation, and number of non-NaN data after a mean command: **mean, stddev, counts;**

The slope, intercept, standard errors, and number of non-NaN data after a trend command: **slope, intercept, errslope, errintercept, counts.**

Some other commands also reset **counts**.

Procedures. The user can write short programs, or 'procedures' consisting of sequences of Scientist's Helper commands, variable definitions and references, and control structures. Procedures are first written in the procedure window or read into Scientist's Helper using the **Read Procedure** item in the File menu. They can then be run using the **execute** command (abbreviated **x**).

The following sample procedure squares column 1, adds it to column 2, and puts the results in column 3:

```
label add
cmath 1 * 1 = 1
cmath 1 + 2 = 3
return
```

This procedure is executed by typing on the last line in the Command Window:

```
x add
```

Procedures can get input from the keyboard. The above example can be modified to ask for a result column:

```
label add
input result 'enter result column'
cmath 1 * 1 = 1
cmath 1 + 2 = @result
return
```

This procedure is executed by typing

```
x add
```

Another way for a procedure to get information is through arguments entered in the command line.

```
label add
cmath 1 * 1 = 1
cmath 1 + 2 = @arg1
return
```

This procedure is executed by typing

```
x add 3
```

(where 3 is the result column).

Procedures can call other procedures:

```
label addAndSquare
x add
cmath 3 * 3 = 3
return

label add
cmath 1 * 1 = 1
cmath 1 + 2 = 3
return
```

Procedures can contain loops. The following procedure plots columns 2, 3, ... against column 1.

```
label plotAllColumns
for column 2 @cols
plot 1 @column
next column
return
```

Procedures can contain conditional statements. The following procedure plots columns 2, 3 ... against 1, querrying each time whether to discuntinue plotting.

```
label plotAllColumns
for column 2 @cols
plot 1 @column
input querry 'continue?  y or n'
if @querry s= 'n'
    return
next column
return
```

Command Syntax: Command keywords are printer in **bold,** arguments in plain text. *Italicized* command words may be omitted.

Essential Commands:

quit

allocate[1] **maximum_rows maximum_columns**

Commands that effect the header:[2]

rows	number_of_rows	
cols	number_of_cols	
title	any_string	
colname	column_number	any_string
interpolated	_ ***true*** **| Ó false**	
samp	sampling_interval	
start	starting_value	

Notes.

1. maximum table size is 4096 by 32, although in practice this is less and depends on the amount of memory on the system.
2. Commands that change the header automatically update the header variables.

Commands for using variables:[1]

setvar variable_name any_string
i*nput* variable_name prompt_string
p*rompt* any_string_1 any_string2 ... any_string_5
t*ype* **v*ariables***
delete **variable** variable_name

set variable_name
- **table** row col
- **|colname** col
- **Ó coefficient**[2] n

vm*ath* input_value_1 **| - | * | / Ô** input_value_2 = variable_name

vf*unction* **sin | cos | tan | asin | acos | atan | sqrt | ln | exp | erf | erfc** input_value variable_name

concat variable_name any_string_1 any_string_2 any_string_3

Notes:
1. A variables name (any string containing no blanks) is distinct from its value, which is its name preceeded by the symbol @.
2. The coefficients are those determined by least-squares methods using the polyfit and multifit commands.
3. The row function returns the row number of a value in col 1 of an interpolated table, based on the current value of samp and start. The row is always in the range 1≤row≤rows.

Graphics Commands:

clear[1]

			_ ***solid***	_
			| dotted	**|**
			| dashed	**|**
plot	abcissa_col	ordinate_col	**| bold**	**|**
			| dots	**|**
			| circles	**|**
			| crosses	**|**
			| stars	**|**
			Ó x	**Ô**

scale[2]	abcissa_col	ordinate_col
xaxis	minimum_value	maximum_value
yaxis	minimum_value	maximum_value
cursor[3]	any_prompt_string	
axes[4]		

Commands for using procedures:

execute[5]	label
sleep	number_of_seconds

Notes.

1. Clears (erases) the graphics screen.
2. Sets the xaxis and yaxis values on the basis of the data in two columns.
3. Used to pick values off of a plot using the mouse. The variables xpos and ypos are set to the cursor position when the mouse is clicked.
4. Plots axes.
5. Automatically updates variables arg1, arg2, arg3, arg4.

Control commands to be used within procedures:

label		any_string			
for[1]	variable_name	starting_integer	ending_integer	increment	
next[1]	variable_name				
goto	label				
return					
if[2]	any_string_1	**s=** **s<>** **=** **<>** **<** **>** **<=** **>=**	any_string_2	*num_of_lines*	

Notes.

1. The **for**-**next** statement pair are used for loops, the **for** statement being placed at the beginning of the loop and the **next** being placed at the end. Loops can be nested.
2. The if statement permits the following num_of_lines to be executed only if the comparison is true. The comparisons s= and s<> test the ascii representation of strings. The comparison =, <>, <, >, <=, and >= c assume that the strings contain numbers and tests the value of these numbers.

Commands for column arithmetic and related operations:

```
                          _ +     _
cmath   input_col_1  | -    |  input_col_2   =   output_col
                          | *     |
                          Ó /     Ô

                          _ #+    _
cmath   constant          | #-    |  input_col   =   output_col
                          | #*    |
                          Ó #/    Ô

                          _ +#    _
cmath   input_col         | -#    |  constant    =   output_col
                          | *#    |
                          Ó /#    Ô

                        _ sin     _
                        | cos     |
                        | tan     |
                        | asin    |
cfunction               | acos    |   input_col   output_col
                        | atan    |
                        | sqrt    |
                        | ln      |
                        | exp     |
                        | erf     |
                        | erfc    |
                        Ó row[1]  Ô

constant[2]             value     col_number   first_row   last_row

                        _ type        input_col                 _
mean[3]                 | keep        input_col    output_col   |
                        | remove    input_col    output_col     |
                        Ó compute   input_col                    Ô

                _ type          input_col_1    input_col_2                 _
trend[4]      | keep            input_col_1    input_col_2   output_col    |
                  | remove      input_col_1    input_col_2   output_col    |
              Ó compute       input_col_1      input_col_2                 Ô
min[5]                  col_number
max[5]                  col_number
sort                    col_number
```

		_ type	input_col_1	input_col_2		_
polyfit[6]	order	**| keep**	input_col_1	input_col_2	output_col	|
		| remove	input_col_1	input_col_2	output_col	|
		Ó compute	input_col_1	input_col_2		Ô
	_ type	indepdent_col_list		dependent_col		_
multifit[7]	**| keep**	indepdent_col_list		dependent_col	output_col	|
	| remove	indepdent_col_list		dependent_col	output_col	|
	Ó compute	indepdent_col_list		dependent_col		Ô

Notes.

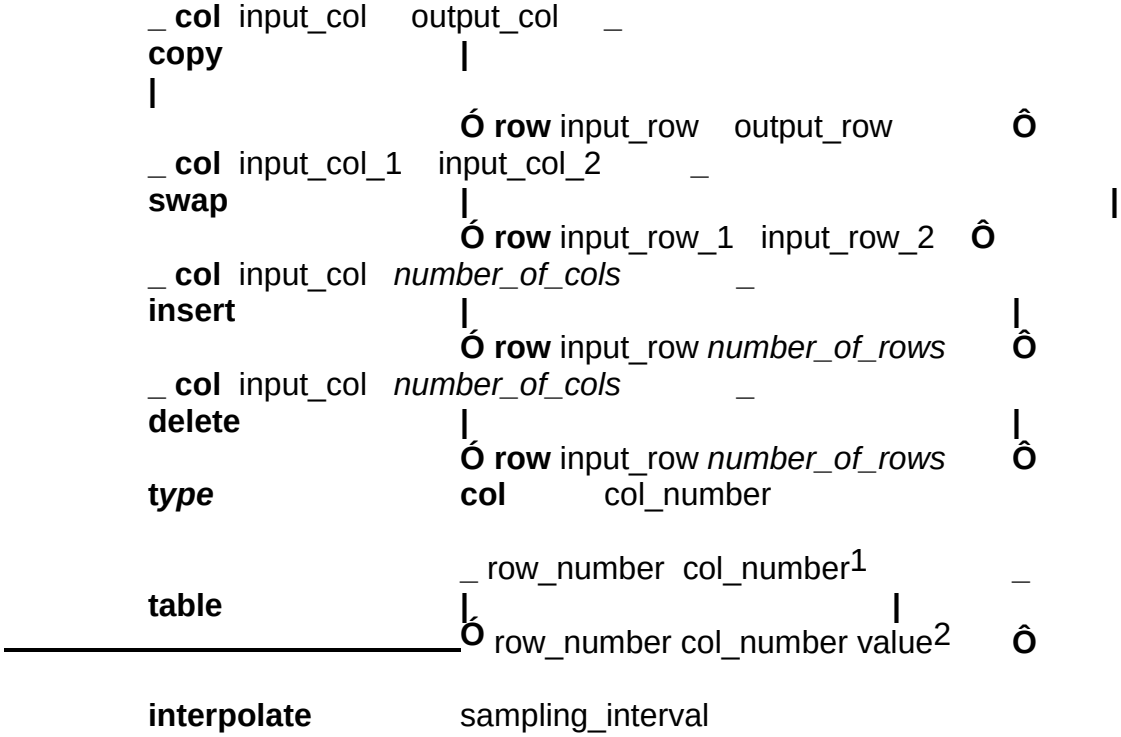
1. The row function returns the row number of a value in col 1 of an interpolated table, based on the current value of samp and start. The row is always in the range 1≤row≤rows.
2. The constant command sets the rows of a column between starting_row and ending_row to the given constant value. If the row limits are omitted, they default to 1 and rows respectively.
3. The mean command computes the mean of a row and updates the variables mean, stddev and counts (the number of non-NaN column entries). 'type' types the result on the terminal, 'keep' puts it in a column, 'remove' subtracts it from a column, and 'compute' has no action except updating the header variables.
4. Trend computes a least-squares fit between two columns and updates the header variables slope, intercept, errslope, errintercept, counts. See note 3 for explanation of keywords.
5. The commands min and max automatically update the variables min and max.
6. Least-squares fit of polynomials of order in range 1-6. This command resets the variable counts. Variables can be set to the values of the coefficients with the set command.
7. Least-squares linear multivariate regression. This commandn resets the variable counts. Variables can be set to the values of the coefficients with the set command. The independent column list consists of columns numbers separated by commas, spaces, or tabs. If spaces or tabs are used, the list must be surrounded by quotes. For example:
multifit type 1,2,3,4 5 6

Time series analysis commands:

band*pass*[1] low_frequency high_frequency input_col output_col
integrate[2] x_col _number y_col _number output_col_number
diff*erentiate*[3] x_col _num y_col _num output_col_num
histogram minimum_value maximum_value number_of_bins
sum[4] input_col_number couput_col_number
spectrum _ **amplitude** _ | **power** | Ó **phase** Ô
taper[5] _ **ascending** x_col output_col start_row end_row _ | | Ó **descending** x_col output_col start_row end_row Ô
conv*olve*[6] input_col# operator_col# operator_length output_col#
noise mean standard_deviation output_col_num

Notes.
1.Second order Chebyshev recursive filter.
2. Integration, $\int_0^x y(x')\,dx'$ by trapezoidal rule.
3. Differentiation dy/dx by first order finite differences.
4. Running sum of input column values.
5. Cosine taper, cos(x) that rises from zero to one or falls from one to zero.
6. Brute-force convolution; operator length better be short. Points off the beginning of the input column are assummed to be zero.

Commands for table manipulations:



Notes:
1. Writes the current table value in the command window
2. Sets the given table entry

Helpful Hints:

1) **Allocate** a table large enough for all your needs, right at the beginning of a Scientist's Helper session. Then, if you want to work with a smaller table, just declare it to be smaller using the **rows** and **cols** commands. Doing this usually speeds things up, since re-allocating space is time consuming.

2) Decreasing the active size of a table with the **rows** and **cols** command does not actually distroy any data. Thus one can always recover the data by increasing the table size. This fact allows one to merge two tables stored in two different files: **allocate** a large table with enough columns to hold both tables, **read** in the first table, increase the number of actiove columns with the **cols** command and **copy** the columns to the right hand part of the table. The **read** in the second table and increase the table size using the **cols** command, thus recovering the data from the first table.

3) Material from the Command Window can be **copied** and **pasted** into the procedure window, and quickly edited into a short procedure.

4) When reading as **ascii table** into MacWrite, choose the option wherby MacWrite interprets carriage returns as paragraphs. Then put enough tabs in the ruler so that all the columns line up properly.

5) If you transfer data to the MacIntosh from another computer using MacTerminal, you will lose any tabs between the columns (MacTerminal converts them to spaces). Therefore, don't send tabs, send one space between each entry. Then use MacWrite to change each space to a tab. You can'y type a tab into the change dialog box, but you can paste it in.