Introduction to spj

spj is an odd combination of spreadsheet program and MINITAB. It is at its best when you want to perform exactly the same set of operations on many different "rows" of data, e.g., if you want to calculate the error for each of several data points. Like plot, spj uses the prompt "*" to request, from the user, commands for action.

A typical spj session begins by using the editor to create a data file. Follow the usual convention in creating the data file: reserve the first line of the file for an identifying sentence, follow that line with columns of data separated by tabs or spaces or commas.

Once you have created the file, start spj by typing

```
% spj
```

(At any time you may exit spj by holding down the *Ctrl* key and simultaneously hitting *D*.) SET spj's FILE variable equal to the filename of the file that contains your data.

```
* SET FILE=filename
```

You now must READ your data file into spj's internal data storage area. spj's internal storage area is divided up into 30 columns of data, with at most 512 rows of data in each column. (If you need more rows, contact Dr. Kirkman for some tricks.) Thus you must specify where each column of your file belongs in spj's memory. To read the first four columns of data in your file into spj's first four columns of memory type:

```
* READ C1-C4 TO C1-C4
```

Other arrangements might also be made. For example,

```
* READ C4,C1-C3 TO C11-C14
```

places the fourth column in the file into spj's eleventh column and file-columns one to three in spj-columns twelve through fourteen. Notice that no data are displayed on the screen. spj displays data only if you request it, e.g.,

```
* PRINT C11-C13
```

will display data on the terminal—the first column on the screen will be spj's internal C11. To WRITE a file from spj's memory, tell spj the filename you want to create and then ask it to WRITE the columns you want in the file (in the order you want them in the file):

```
* SET FILE=filename
* WRITE C3,C4,C1
```

In this example, C3,C4,C1 is the "clist", i.e., the set of columns you want to appear in the file; the first column in the file will be C3, the second C4, etc.

Typically, you will want to modify the READ in data before you WRITE the data. The most powerful data modifying command is LET Cx=expression. (Unlike most BASICs the LET is *not* optional.) The expression consists of any valid combination of operations (+, -, *, /, ** or ^, note: (-2)**3=$|-2|^3 = 8$), column numbers (C1-C30), constants (e.g., 365.25, 6.02E23), variables (A,B,C,X,Y,Z),

parenthesis, and functions. (All fortran, single-real-argument functions (`abs, acos, asin, atan, cos, cosh, exp, int, log, log10, nint, sin, sinh, sqrt, tan,` and `tanh`) are valid, plus `INORM, K, NORM, PI, RAN, ROW,` and `WINDOW. RAN` produces "random" numbers; `ROW` has a value equal to the row number; `WINDOW(`x`)` has the value 1 iff $a \leq x < b$, 0 otherwise.) Thus

```
* let c4=(-c2+sqrt(c2*c2-4*c1*c3))/(2*c1)
* let c5=(180/pi)*asin(c1/2.03)
```

are valid statements.

There are five commands designed to produce statistical information. `CSTAT` reports sum, mean, standard deviation, maximum and minimum for a single column of data. The other four operate on several columns of data, producing a `MAX, MEAN, MIN` or `SDEV` for each row of data. For example,

```
* MEAN C1-C3,C5 TO C6
```

will, for each row, find the mean of the four specified numbers and put the result in `C6`.

There are three special purpose commands. `FIT` `C`x,`C`y finds a least-squares line through the specified columns of data. `SORT` will sort a single column of data, highest-to-lowest, `TO` another column. `?` is a handy calculator, allowing you to evaluate an expression. So if you need to know the value of $\frac{\pi}{2}$, just type

```
* ?  pi/2
```

and the answer will immediately appear. (Column references in the expression will be evaluated using the `IBEGIN` row.)

Thus, `spj`'s command set includes: `?, CSTAT, FIT, HELP, LET, MAX, MEAN, MIN, PRINT, READ, SDEV, SET, SHOW, SORT,` and `WRITE`.

Perhaps the most useful variable is `CFORM`, which is the fortran real-variable format the program will use to `PRINT` or `WRITE` column data. As may be found in a fortran language reference manual, formats for real numbers usually have the form: [`F, E,` or `G`]width.decimal, e.g., `F5.2`. Thus you can request a shorter width for each real number `PRINT`ed, allowing more real numbers to be displayed. In addition to working with numerical data, `spj` has two columns of 32-character, alpha-numeric data available: `L1` and `L2`. `LFORM` is used to format this literal data. The typical form is `A`width, e.g., `A10`.

You can also `SET` values for the expression variables: `A, B, C, X, Y` and `Z`.

`IBEGIN` and `NROW` determine the first row used in the internal storage area and the total number of rows used. Do not mistake `NROW` and `ROW`. `ROW` is the number of lines of text skipped before the program `READ`s in the columns.

Special Commands

Start: "`% spj`"
`$`*UNIX* command — do any *UNIX* command from inside `spj`
`@`*filename* — execute indirectly a file of `spj` commands
Ctrl D — to exit program
Examples:

```
* $ls *.dat
* @fet.spj
```

spj Commands
(unambiguous abbreviations and lowercase allowed)

`?` expression	immediate calculation, Cx(ibegin) used if Cx value requested
`CSTAT` Cx	report sum, mean, sdev, min, max for column Cx
`FIT` Cx,Cy	report slope, yinter, r for linear least-squares fit Cx=x, Cy=y
`HELP`	print this file
`LET` Cx = expression	assign the value of expression to column Cx
`MAX` clist `TO` Cx	find maximum of columns in clist, put result in Cx
`MEAN` clist `TO` Cx	find mean of the columns in clist, put result in Cx
`MIN` clist `TO` Cx	find minimum of columns in clist, put result in Cx
`PRINT` list	display on terminal the list shown, in order shown
`READ` filelist `TO` spjclist	transfer columns of data in `FILE` to `spj`
`SDEV` clist `TO` Cx	find sdev of columns in in clist, put result in Cx
`SET` variable=value	change internal variable values
`SHOW`	report internal variable values
`SORT` Cx `TO` Cy	sort the single column Cx, put result in Cy
`USER`	link in your own data manipulator
`WRITE` list	write to file the list shown, in order shown

where *column* stores real data (`C30` max); *label* stores word data (`L2` max); Cx, Cy are particular column names e.g., `C21`; *clist* is a list of columns e.g., `C1,C2,C4-C9`; *list* is a list of columns and labels e.g., `L1,C3,C2`; an *expression* is a combination of the operations: `-, +, *, /, **` or `^`, parenthesis, variables (`a, b, c, x, y, z`), constants, functions (`abs, acos, asin, atan, cos, cosh, exp, inorm, int, k, log, log10, nint, norm, pi, ran, row, sin, sinh, sqrt, tan, tanh`, and `window`) and column names.

Examples:

```
* LET C1 = ROW/30*PI
* let c2=(c1-sin(c1))
* let c3=cos(c1)-1
* SET FILE=cycloid.dat
* write c2,c3
```

Variables

`A`	used with `LET`; a named constant in an expression (0)
`B`	same (Note: `A`&`B` are used in function `WINDOW(X)` which has value 1 iff $a \leq x < b$)
`C`	same
`CFORM`	used in `PRINT` and `WRITE`; fortran format for real numbers (g)
`FILE`	used with `READ` and `WRITE`; determines what file is used (data.spj)
`IBEGIN`	used in all stat ops; beginning row number for operation (1)
`LFORM`	used in `PRINT` and `WRITE`; fortran format for label (a23)
`MEAN`	reports most recent mean after `CSTAT` (0)
`NROW`	number of valid rows in program (max 100) (100)
`R`	linear-correlation coefficient for most recent `FIT` (0)
`ROW`	used in `READ`; first valid row of data in file (2)
`SUM`	reports most recent sum after `CSTAT` (0)
`SDEV`	reports most recent sdev after `CSTAT` (0)
`SLOPE`	reports most recent slope after `FIT` (0)
`X`	same as `A`
`Y`	same
`YINTER`	reports most recent yinter after `FIT` (0)
`Z`	same as `A`