

RTABLE

R DATA FRAME SOFTWARE PACKAGE

INTRODUCTION

This HP Prime software is a public domain contribution and may be used for any purpose as long as a traceable reference is made to the original source repository from which it was obtained.

This software was developed and is maintained by Mark A. Fleming. Questions, suggestions and bug reports can be directed to the User Forum of the Museum of HP Calculators, at www.hpmuseum.org or to MoHPC member ID mffleming.

PURPOSE

The functions contained in this program use the Spreadsheet application of the HP Prime to provide an analog of the Data Frame object in the R language and environment for statistical computing and graphics. In particular, the functions provide a means of loading a spreadsheet from a CSV file and saving spreadsheet content to a CSV file. Additional table management functions are provided, for example to add or remove rows or columns.

The functions in this package are used in the context of an HP Prime Spreadsheet application. Due to current limitations on clearing spreadsheet content, a data analytic task should begin in the default Spreadsheet application. Spreadsheet content and working CSV files can be deleted by resetting the application. Once data has been imported and properly formatted, a copy of the Spreadsheet application can be saved as a user application.

For more information on R refer to the project site at www.r-project.org for documentation and software download.

LIMITATIONS

As of HP Prime software release 8151 (2015 6 17) there are certain limitations that affect the functionality and usage of this software package.

- No function is provided to reset a spreadsheet or spreadsheet cell to a default empty state.
- Column and row labels can be programmatically set but not individually queried for value.
- Cell content, including formulas and formatting, can be copied manually to other cells but programmatic capability is less extensive.

The first limitation means that the extent of a spreadsheet cannot be reduced unless the spreadsheet is reset, which clears the application of all content, variables and files. A reset can only be performed for the built-in Spreadsheet application; any saved copy of the application will simply be deleted. In consequence, any preliminary data import and manipulation should be done within the built-in application, and a copy of the spreadsheet saved only when data content is stable.

The second limitation means that row and column names are cached in a structure for quick lookup when needed, such as when saving names as part of a CSV file. Since a column or row name can be updated manually, this structure could become outdated. Using one of the functions that retrieve row or column names will update the structure if you have manually changed a name and now wish to use the name as a reference in a table function.

RTable PACKAGE FUNCTIONS

The RTable program file contains functions that expect to access the built-in Spreadsheet application or one of its saved instances. Efforts to invoke functions or edit the RTable program file in another application context will result in an error.

CSV FILE FUNCTIONS

Three functions, *read*, *write* and *writea*, are provided to load or save spreadsheet content. The program is configured to process files conforming to the RFC 4180 standard. A conforming file contains records delimited by carriage-return/line-feed characters and fields separated by commas. The first record can optionally contain the name of each column in the table.

<code>read(filename)</code>	Read content of specified CSV file into spreadsheet cells according to option settings.
<code>write(filename)</code>	Write content of spreadsheet cells to specified CSV file according to option settings.
<code>writea(filename)</code>	Write content of spreadsheet cells to the end of specified CSV file according to option settings.

The following functions manage options that the CSV functions use.

<code>initOptions()</code>	Create and initialize option variables for the enclosing application.
<code>setstd()</code>	Set options that configure CSV file operations to conform to the RFC standard.
<code>set(option,value)</code>	Set a specific option to an appropriate value. Equivalent to <code>AVars(option) := value</code>
<code>get(option)</code>	Return the value of a known option. Equivalent to <code>AVars(option)</code>

CSV FILE OPTIONS

Options are provided to change the record delimiter character, and to process files in which fields are delimited by fixed column widths. The table below shows options that affect file processing.

Option	Default	Description
<code>style</code>	"delimited"	Determines whether a CSV file has fields separated by a delimiter character ("delimited") or by fixed column widths ("fixed").
<code>separator</code>	","	Character used to delimit fields in a delimited record. Can be set to other characters, such as the ";" character.
<code>columns</code>	{1}	Sets the column stops for each field in a record for fixed-width CSV files.
<code>namecols</code>	0	True/False flag indicating that the first record of a CSV file contains the names of table columns that <code>read()</code> processes, and that <code>write()</code> will output column names as the first CSV file record.
<code>namerows</code>	0	True/False flag indicating the first field in a record (except column name header record) is the row name.
<code>quoted</code>	0	Indicates the <code>write()</code> function should output text fields surrounded by the quote character. The <code>read()</code> function will strip enclosing quotes and properly process embedded quotes.

Additional options are used or set by the *read* and *write* functions. Of particular importance are the bounding variables `rowstart`, `rowend`, `colstart` and `colend`. The bounding variables define a subset of the Spreadsheet to which certain functions will be applied. The start values default to one. The end values are set following a call to the *read* function. When using the *write* function, be sure the bounding variables delimit the section of the spreadsheet you wish to save. Setting the `rowend` or `colend` variables to -1 will use the maximum row/column extent of the spreadsheet as the boundary.

Option	Default	Description
<code>attribute</code>	0	Cell attribute value to read or write, from the <code>Cell()</code> attribute list (i.e. 0 for value, 1 for formula, 2 for name, etc.)
<code>rowstart</code>	1	Table starting row for a read or write operation, allowing a subset of a table to be written or a table to be offset from the origin of the spreadsheet.
<code>rowend</code>	-1	Set to last row read by the <code>read()</code> function, or used to subset the content written to a CSV file by the <code>write()</code> function.
<code>colstart</code>	1	Table starting column for a read or write operation, allowing a subset of a table to be written or a table to be offset from the origin of the spreadsheet.
<code>colend</code>	-1	Set to last column read by the <code>read()</code> function, or used to subset the content written to a CSV file by the <code>write()</code> function.

R TABLE FUNCTIONS

The following functions have a rough correspondence with methods or functions associated with the R data frame object. Function naming is consistent with the idea of providing an environment that is a close match to R.

<code>nrow()</code>	Returns the number of rows in the current bounded table.
<code>ncol()</code>	Returns the number of columns in the current bounded table.
<code>bounds()</code>	Returns the list <code>{rowstart, rowend, colstart, colend}</code> that defines the boundary of the current table.
<code>setbounds(blist)</code>	Accepts a list returned by the <code>bounds()</code> function to set a new current table boundary.
<code>merge(b1,b2)</code>	Alters the bounding variables so that they encompass the boundary of two tables delimited by their bounds list. Equivalent to $\{ \min(b1(1),b2(1), \max(b1(2),b2(2)), \min(b1(3),b2(3)), \max(b1(4),b2(4)) \}$
<code>names()</code>	Returns a list of column names, delimited by the <code>colstart</code> and <code>colend</code> bounding variables. Unnamed columns are marked by the empty string.
<code>rownames()</code>	Returns a list of row names, delimited by the <code>rowstart</code> and <code>rowend</code> bounding variables. Unnamed rows are marked by the empty string.
<code>setnames(namelist)</code>	Sets the name of columns between <code>colstart</code> and <code>colend</code> to the strings contained in the list argument. Names are truncated to the internal eight character limit.
<code>setrownames(namelist)</code>	Sets the name of rows between <code>rowstart</code> and <code>rowend</code> to the strings contained in the list argument. Names are truncated to the internal eight character limit.
<code>rfactor(column)</code>	Returns the sorted, unique list of values in a column between the bounding <code>rowstart</code> and <code>rowend</code> rows. A column is designated by its name or number, numbering from the first column as column 1.

When the idea of a table is mentioned in this document, we mean the R definition of a collection of tabular data where all rows have the same number of fields and all fields in a column have the same data type and meaning. The implementation of a table in an HP Prime spreadsheet is that of a bounded set of rows and columns within a sheet. This means a sheet can hold more than one table, as long as we keep track of the table boundaries. **Keep in mind however that row and column names or numbers refer to the absolute spreadsheet row or columns, not the relative table row or column.**

The origin of a table need not be the first row and column, so the bounding variables `rowstart`, `rowend`, `colstart` and `colend` are used to define a table's extent within a sheet. Two functions are provided to set or retrieve all four of these variables as a list. A third function (`merge`) allows you to redefine the boundary region so that it encompasses two tables, causing the bounding variable values to be updated accordingly.

For the `read` function, if the `namecols` option is true but a table does not begin in row one, the corresponding spreadsheet column names will not be changed. Likewise, row names including in a CSV file will be discarded by `read` if the table does not begin in column one. Row and column names will be included in the output CSV file if the corresponding options `namecols` or `namerows` is set to true. This interpretation of table boundary and naming allows a subset of a larger table to be read or written without impacting the larger, enclosing whole.

TABLE SUPPORT FUNCTIONS

Additional functions are provided to manage a table and its row and columns.

<code>clear()</code>	Clears all cells bounded by <code>rowstart/rowend</code> and <code>colstart/colend</code> variables. If <code>rowend</code> and/or <code>colend</code> are set to -1, then the last occupied row/column of the spreadsheet is used.
<code>clearsheet()</code>	Clears the entire spreadsheet by clearing each cell from origin to the outer boundaries.
<code>getrow(row)</code>	Returns cell content of the specified row between <code>colstart</code> and <code>colend</code> as a list. Uses the <code>attribute</code> option when reading cell content. Row may be specified by its name in quotes.
<code>getcol(column)</code>	Returns cell values of the specified column between <code>rowstart</code> and <code>rowend</code> as a list. Uses the <code>attribute</code> option when reading cell content. Column may be specified by its name in quotes.
<code>setrow(row,vlist)</code>	Sets the cells between <code>colstart</code> and <code>colend</code> of the specified row to the supplied list of values. Uses the <code>attribute</code> option to determine what is set in a cell. Row may be specified by its name in quotes.
<code>setcol(column,vlist)</code>	Sets the cells between <code>rowstart</code> and <code>rowend</code> of the specified column to the supplied list of values. Uses the <code>attribute</code> option to determine what is set in a cell. Column may be specified by its name in quotes.
<code>rmrow(row)</code>	Removes the absolute row number cell contents between <code>colstart</code> and <code>colend</code> , shifting rows below up one. Row may be specified by its name in quotes.
<code>rmcol(column)</code>	Removes the absolute column number cell contents between <code>rowstart</code> and <code>rowend</code> , shifting columns after to the left. Column may be specified by name if the row is named.
<code>insrow(row)</code>	Inserts a row into the current bounded table, shifting all other rows below the table downward. Row may be specified by name if the row is named.
<code>inscol(column)</code>	Inserts a column at the specified absolute column position. The column may be specified by name if the column is named.

The HP Prime Spreadsheet application provides means to manually clear rows or columns, and copy data between rows or columns. No means are provided for programs to perform these cell-aggregate operations, so RTable supplies some useful table manipulation functions. Functions are provided to insert a row or column, delete one or more rows or columns, and read or write the cell contents of a row or column. All of these operations are carried out on the cells that fall within the area of the table bounding variables.

Once again, the bounding variables influence the operation of this class of functions. When rows or columns are inserted or deleted, the operation only affects those row or column cells within the current table boundary. For example, if two tables are stored vertically in the spreadsheet and a column is deleted from the active table, the corresponding column is unaffected in the other table. In order to impact both tables with an operation, the active boundary would need to be extended using the *merge* function.

The *getrow*, *getcol*, *setrow* and *setcol* functions apply to an absolute spreadsheet row or column number. A column or row name can be used in place of a number. The name of a column can also be the default one or two letter designation, as in "C" or "BG" (names are case insensitive). The delete row and column functions can accept a list of names or numbers, while the insert functions only accept a single name or number.

One last observation is important concerning table boundaries and the bounding variables. Row names and column names are only affected by insert or delete operations if the table boundary includes the corresponding first column or first row. For example, deleting a column from a table that starts in row one will delete the corresponding column label and shift all labels on the right by one column to the left. On the other hand, doing the same with a table that does not start in the first row will have no effect on column labels. One way to remember this is; if a table touches the column or row names, the names are part of the table.

EXAMPLES

The following examples illustrate the way in which the RTable program can be used to manage spreadsheet data.

EXAMPLE 1: MOVING FILES BETWEEN APPLICATIONS

Each application has its own namespace for variables and files. You may wish to share data between spreadsheet applications or use an application as a repository of data files. Let's suppose we want a repository that we can pull files from or save results to. We can use the Connectivity Kit to insure the repository is backed up and files are kept safe.

First, create a repository application by selecting the Spreadsheet built-in application and pressing the Save softkey. Let's name the application CRAN, after its R counterpart. Once the HP Prime is connected to the Connectivity Kit, the repository will appear in the Application list. Open the application list, and then open the CRAN folder to see the Files folder. You can right-click on that folder and select Add to open a file dialog. After copying files to the folder, right-click the CRAN folder and select Send to Class. Your files will be transferred to the HP Prime. To pull new or updated files back to the Connectivity Kit, simply right-click the Calculator and select Refresh. Alternatively, highlight your CRAN application from the App View and press the Send softkey.

Copying files between applications is a simple matter of reading the file content, as a list, into an intermediate variable and then writing that content to another file. Assume you are in the Spreadsheet application and want to load one of the files you placed in the CRAN repository. Use the commands in Home shown here to effect the transfer.

```
CRAN.AFiles()  
L1 := CRAN.AFilesB("sample.csv", 0, AFilesB("sample.csv"))  
AFilesB("mysample",0) := L1
```

The first line above returns a list of files in the CRAN repository so that we can pick out the one we wish to access. The second line reads the content of the "sample.csv" file into the list variable L1 (any local variable can be used). The third line writes the content of the repository file to a local file called "mysample" (there is no real need for a ".csv" extension, is there?)

EXAMPLE 2: LOADING A CSV FILE

The RTable *read* function will open and parse a local Spreadsheet application file, storing the file content into spreadsheet cells. The format of the file at minimum must match the RFC 4180 standard. Each line of the file is a record, terminated by carriage-return/linefeed (RTable is fairly relaxed about record termination). Each record contains one or more fields separated by a comma. Each field can be a number or a text string. Quotes around text are optional and may be used when the text contains an embedded separator character.

RTable extends the standard by allowing the field delimiter to be something other than a comma (*separator* option), and by allowing fields to be defined by starting column (fixed width fields). To load a file into the spreadsheet, set options as appropriate before calling the read function. You can clear the spreadsheet of all content by using the *clearsheet* function, and set options according to the RFC using the *setstd* function. The following lines load a file in standard CSV format where the first record contains the names of each record column.

```
clearsheet();  
setstd();  
set("colnames", 1);  
read("sample.csv");
```

In this example, a CSV file with four fields per record in fixed column format is loaded. The fields start in columns 1, 10, 16 and 25. Columns are unnamed.

```
setstd();
set("style", "fixed");
set("columns", {1,10,16,25});
read("fixed.csv");
```

EXAMPLE 3: SAVING A CSV FILE

Saving a spreadsheet in CSV format is similar to loading a file into the spreadsheet, but the bounding variables need more attention. When loading a file, the bounding variables `rowstart` and `colstart` determine where records and their fields begin to be inserted into the spreadsheet. When an application is reset or the `clearsheet` function is used, these variables default to 1. The `read` function sets the other bounding variables `rowend` and `colend` after loading the CSV file.

If the extent of the spreadsheet content is altered using `RTable` functions to insert or delete rows or columns then the bounding variables are updated to reflect change. However, if manual changes are made then you need to update the two endpoint variables to reflect the true extent of the content that you wish to save. The bounding variables can also be used to subset the content of that you want saved to a file. The `write` function will also allow you to create a CSV file using a different delimiter or even using fixed column output.

As with the `read` function, you can configure a different separator character or output the spreadsheet cells as a fixed column CSV file. You also have the option of enclosing text values in quotes.

```
setstd();
set("colnames", 1);
set("rownames", 1);
write("sample.csv");
```

Note that the `rownames` option has been set `True` in order to have the row name in the first field of each record. Unless you have actually named the rows, the row number will be used for the name. R generally expects numbered records when reading a CSV file, and having your records numbered may also prove useful in certain other contexts.

EXAMPLE 4: MANAGING MULTIPLE CSV FILES

Not only can the bounding variables be used to delimit the content of what you save to a CSV file, they can also be used to load or save multiple files to the same spreadsheet. The `bounds` and `setbounds` functions can be used to manage several separate tables in the same spreadsheet. Below is an example of loading two separate files for processing within the same spreadsheet. The second file is loaded beneath the first, with an offset of a few blank rows.

```
clearsheet();
setstd();
set("colnames", 1);
read("part1.csv");
table1 := bounds();
set("rowstart", get("rowend")+5);
set("colnames", 0);
read("part2.csv");
table2 := bounds();
setbounds(table1);
```

Following the above set of commands, each table can be processed separately as if they were the only ones occupying the spreadsheet. The *setbounds* function can switch between each table and further operations will only affect the cells within each of the table boundaries.

IN CLOSING

At the end of RTable is the exported *regressiontest* function that might provide some examples of using the RTable program. Feel free to delete it if the space is needed!

Some shortcomings are inevitable in a first release. The version 1.0 code allows you to specify which attribute of a cell to read or write via the `attribute` option, but the read function does not correctly set the formula as yet. The initial design goal was to simplify fixed column output by using the widest element of each column to define column width. For now though, you will need to supply the starting column number for all table fields.