DataTools 1.6

Ron Doerfler

14 April 2019

The DataTools program provides a set of tools for preparing data in a CSV or tab-delimited text file prior to loading the data into Excel or other data analysis tools. Using these data tools is generally much faster and more convenient than opening data files directly in Excel and manually performing the operations, particularly for large files or for files too large to load into Excel. In fact, these tools are excellent alternatives at times to working with Excel generally. The latest release can be downloaded from http://www.sliversoftware.com/download.htm.

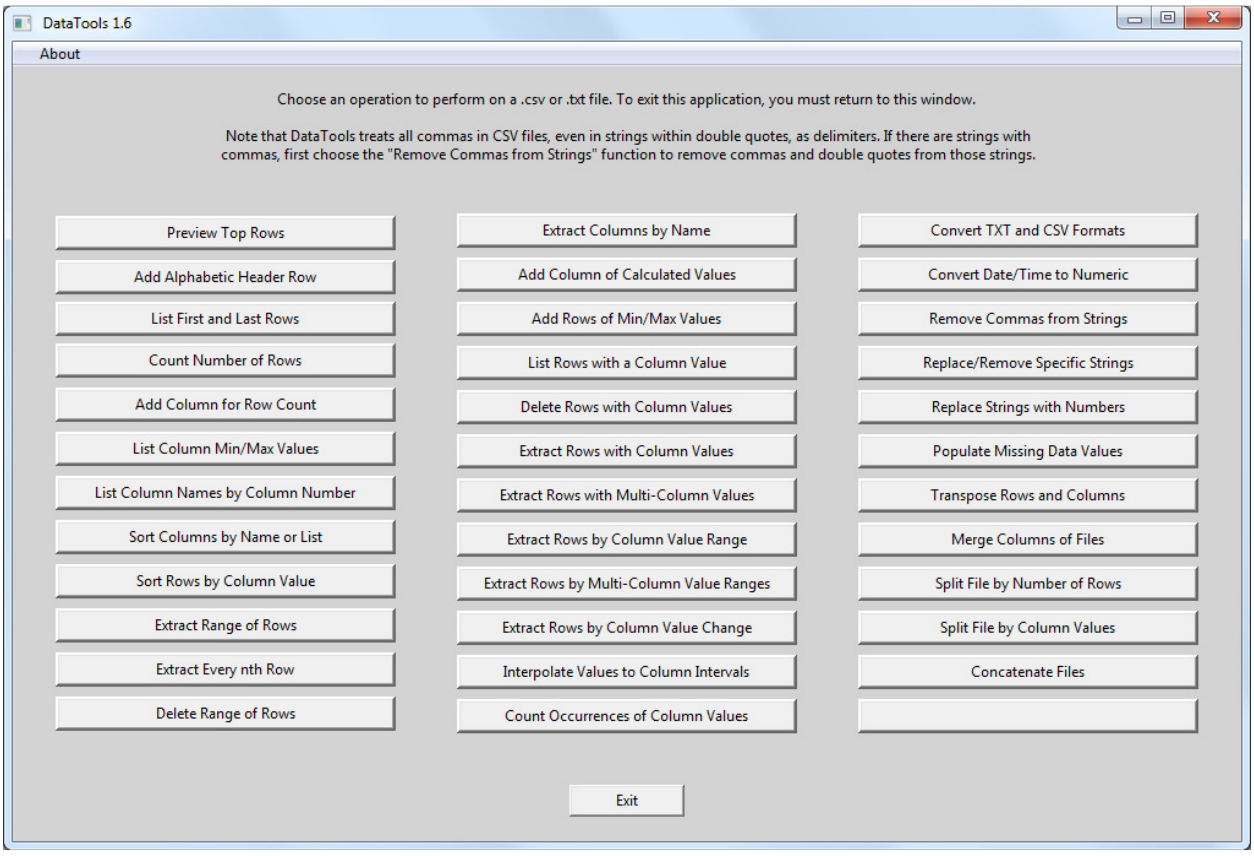

Figure 1. Tool Selection Window.

Selecting a tool from the main Tool Selection Window in Figure 1 produces a new window that describes the operation of that tool and the required input fields, as well as listing the first 25 lines of the output file when the operation is completed. An example is shown in Figure 3. The type of file (CSV or tab-delimited TXT) is determined by the input file suffix, with a default of CSV if a different suffix occurs. The tools operate on a data file with or without a header row, depending on a checkbox setting. For tools that have the option of choosing variables (such as *Extract Columns by Name*), the selected variables can also be saved to a text file or read from a text file that lists one variable per line (see Figure 2).

Note that DataTools treats all commas in CSV files, even in strings within double quotes, as delimiters. If there are strings with commas, first choose the *Remove Commas from Strings* function to remove commas and double quotes from those strings

The functions provided by the DataTools program are:

1. **Preview Top Rows** – Display the top 25 rows of an input file

2. **Add Alphabetic Header Row** – Add a header row to a file of pure data so the columns have variable names. An alphabetic scheme such as in Excel columns (A-Z, AA-AZ, etc.) is used. This function may also be needed in order to perform other functions that involve selecting specific columns.

3. **List First and Last Rows** – View the first and last rows of one or more files, with an option to select which column variables to display the data for assuming all the files have the same header row. This is very useful when a large file has been split into smaller files and you need to find out which file contains a particular value of a variable that is sequential, such as time or a frame counter. There is also an option to show the row numbers in the output, which is useful for determining the number of rows in a file.

4. **Count Number of Rows** – Display the number of data rows of one or more input files.

5. **Add Column for Row Count** – Insert a new first column with a specified variable name that contains the row number, with the first data row being row 1. This is useful, for example, as the variable to assign to the x-axis of a scatterplot in order to plot a variable against the sample or time.

6. **List Column Min/Max Values** – List the minimum and maximum values of columns of data from the file based on one or more selected variable names.

7. **List Column Names by Column Number** – List the column number of every column name in the input file, either sorted alphabetically by column name or in numerical order. This is useful when columns need to be selected by column number, such as when plotting data in Matlab.

8. **Sort Columns by Name or List** – Sorts the columns of the input file either by algorithm (ASCII or dictionary sorting in increasing or decreasing order) or by a list of column (variable) names in a text file that is selected. In the latter case the names are listed one per line in the same format as when variable selections are saved in Sliver windows. Column names that are not listed are not saved in the output file. A warning is provided if any column names listed in the file do not exist in the input file.

9. **Sort Rows by Column Value** – Sort by the value in a selected column. Sorting can be performed in Numeric, ASCII or Dictionary order, either increasing or decreasing.

10. **Extract Range of Rows** – Extract a range of rows between start and end values entered here and store them into an output file.

11. **Extract Every nth Row** – Extract every nth row and store them into an output file (also called *decimating* a file).

12. **Delete Range of Rows** – Delete a range of rows between the start and end values entered here and store the result into an output file.

13. **Extract Columns by Name** – Extract columns of data from an input file based on one or more selected variable names (see Figure 2) and then save them in an output file.

14. **Add Column of Calculated Values** – Add a user-named variable column with values calculated from one or more existing columns using math functions including arithmetic, powers, roots, and logarithmic and trigonometric functions. The calculation formula can be saved to a text file or read in from a text file. Since the formulas refer to columns by number, a list of column numbers for each column name is displayed for reference when an input file is selected.

15. **Add Rows of Min/Max Values** – Add two rows to the top of the data (after the header row), the first containing the minimum values of the columns of data and the second containing the maximum values. Selecting any single variable in the variable selection window will populate the rows with the minimum and maximum values of each individual column. Selecting more than one variable will produce rows that will share the common minimum and maximum values of these selected variables, while the other values will remain the minimum and maximum values of the individual columns.

    This function is useful to set automatic ranges of a plot such as a parallel coordinate plot such that the axes of the selected variables all have the same range. The two additional rows will appear as extra data points/lines at the ends of the ranges in the plots and may be deleted after plotting the data.

    NOTE: Because two rows of min/max values are added at the top of the output file, do not use this output file for statistical analysis of the data.

16. **List Rows with Column Value** – List all rows for which a selected variable has the entered numerical value, or the rows with the nearest value to the entered value.

17. **Delete Rows with Column Values** – Delete all rows for which one or more selected variables have the entered numerical value or string.

18. **Extract Rows with Column Values** – Extract all rows for which one or more selected variables have the entered numerical value or string.

19. **Extract Rows with Multi-Column Values** – Extract all rows for which one or more selected variables satisfy Boolean combinations of numerical values or strings across multiple variables.

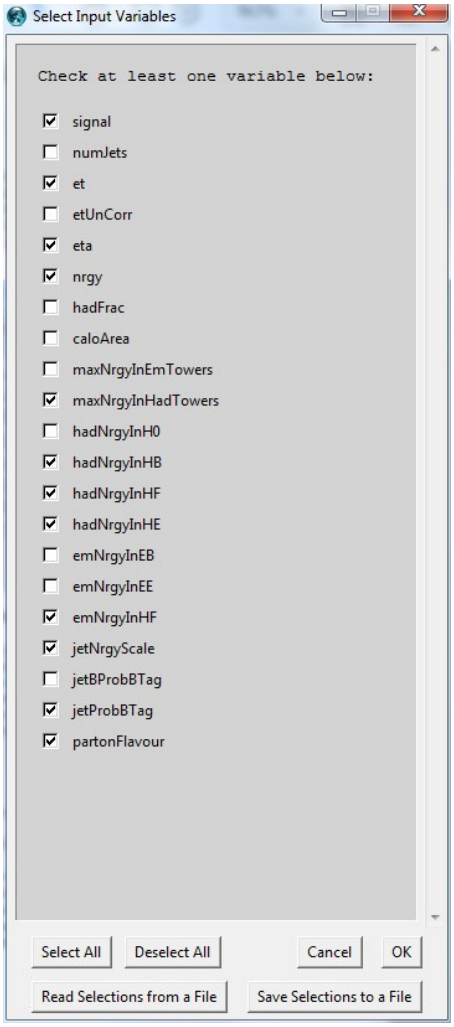

Figure 2. Variable Selection Window.

20. **Extract Rows by Column Value Range** – Extract all rows for which the numerical values of the chosen variable lie within specified minimum and maximum values. A screenshot of this function is shown in Figure 3.

21. **Extract Rows by Multi-Column Value Ranges** – Determine for each row whether the numerical values of one or more selected variables lie within specified minimum and maximum values for those variables. An AND or OR function is selected to extract rows that either meet the range requirements for all the selected variables or any one of them.

22. **Extract Rows by Column Value Change** – Extract the first row and all rows for which the numerical value or string of a selected variable changes. A new column is added on the left to describe the change, such as "0 to 1".

23. **Interpolate Values to Column Intervals** – Interpolate all row values in the input file to fixed intervals of a selected column to aid in syncing data measured asynchronously, where this function supports time intervals as well as value intervals--for example, interpolating rows to intervals of a time variable ranging from 08:40:00 to 10:20:30 in increments of ten seconds (:10). If the column selected for the fixed intervals

is already sorted in increasing order in the input file, there is an option to skip the sorting process to speed the process.

24. **Count Occurrences of Column Values** – Count unique values or strings in a selected column of the input file and save the values or strings, counts and count percent of the total as columns in the output file. There is an option to add the value in another selected column to the count rather than increasing the count by 1 for each row with the value. There are also options to sort the rows in the output file by the value or string (as a dictionary sort) or by the count (in decreasing order). An example case for using this function is processing a CSV file of bird sightings downloaded from eBird.org, where each sighted bird is listed in a separate row and the total sightings for each species is desired. There is another column in that dataset for the number observed of the bird, and here the option to increase the count by that variable rather than by 1 is useful to get the actual total count for each bird species.

25. **Convert TXT and CSV Formats** – Convert a tab-delimited TXT file to a comma-separated CSV file, or vice-versa.

26. **Convert Date/Time to Numeric** – Convert standard date and time formats, which can include time zone mnemonics, in user-selected columns to numeric values, which can then be plotted. The output format can be chosen as the number of days (including the fractional amount) since the start of the oldest date across all selected columns, or seconds relative to the start of today (which is also useful for time intervals such as 01:24:00).

    Acceptable input formats include hh?:mm?:ss?? with or without mm/dd?/yy? or dd monthname ?yy? or day or dd monthname yy or ?CC?yymmdd or ?CC?yy-mm-dd or dd-monthname-?CC?yy. So, for example, *10 December 2014* or *12/10/14* or *12/10/2014* or *141210* or *14-12-10* or *01:32:45* or *01:32:45 12/10/14* are allowed, but for a CSV file the comma delimiter cannot be used, disallowing *December 10, 2014*. If only a time is specified (hh?:mm?:ss??), the current date is assumed. If the string does not contain a time zone mnemonic, the local time zone is assumed. The default year is the current year. If the entered year is less than 100, the years 00-68 are treated as 2000-2068 and the years 69-99 as 1969-1999. Time relative to the current time, such as *4 hours* is also converted, with acceptable units of *year*, *fortnight*, *month*, *week*, *day*, *hour*, *minute* (or *min*), and *second* (or *sec*). The relative units can be singular or plural, as in *3 weeks*, and can be modified with *tomorrow*, *yesterday*, *today*, *now*, *last*, *this*, *next* or *ago*.

27. **Remove Commas from Strings** – Remove commas and any surrounding double quotes from a CSV file. DataTools associates commas in double quotes in CSV files as delimiters, so this function exists to remove any commas in double quotes, as well as the quotes, from a CSV file prior to performing other operations.

28. **Replace/Remove Specific Strings** – Replace any string or substring in selected columns of an input file with another string or number (or blank to delete the string).

29. **Replace Strings with Numbers** – Most data analysis tools only supports numerical data. When a file has variables that contain strings, this function assigns unique numbers to them. Along with the new output file it also creates a text file that lists the string mapping to the numbers. Certain categorical strings are automatically assigned logical values of 1 and 0 if available for that variable (such as True and False, Yes and No, On and Off, Low and High).

30. **Populate Missing Data Values** – Sometimes data is simply unknown for certain data samples and the value is left empty. Most data analysis tools will not allow such a file to be loaded. This function has options to fill all missing data in a file with a given value or to linearly interpolate between the known values for each variable.

31. **Transpose Rows and Columns** – Transpose the rows and columns of an input file and store the result into an output file. A header row in the input file will accordingly lie in the first column of the output file.

32. **Merge Columns of Files** – Horizontally (left-right) merge columns from two different files into an output file. The rightmost set of columns has a user-specified prefix added to the variable names in case the two

files contain the same kind of data. This function is useful to analyze two sets of data together whose data are row-aligned or have been made row-aligned by deleting rows.

33. **Split File by Number of Rows** – Split a file into multiple files having a given maximum number of lines. This is useful when a file has too many rows to analyze as a group.

34. **Split File by Column Values** – Split a file into multiple files based on distinct strings or values in a selected column.

35. **Concatenate Files** – Choose up to 20 files and the order in which to concatenate them into an output file. For greater numbers of files, repeat using previously concatenated files.

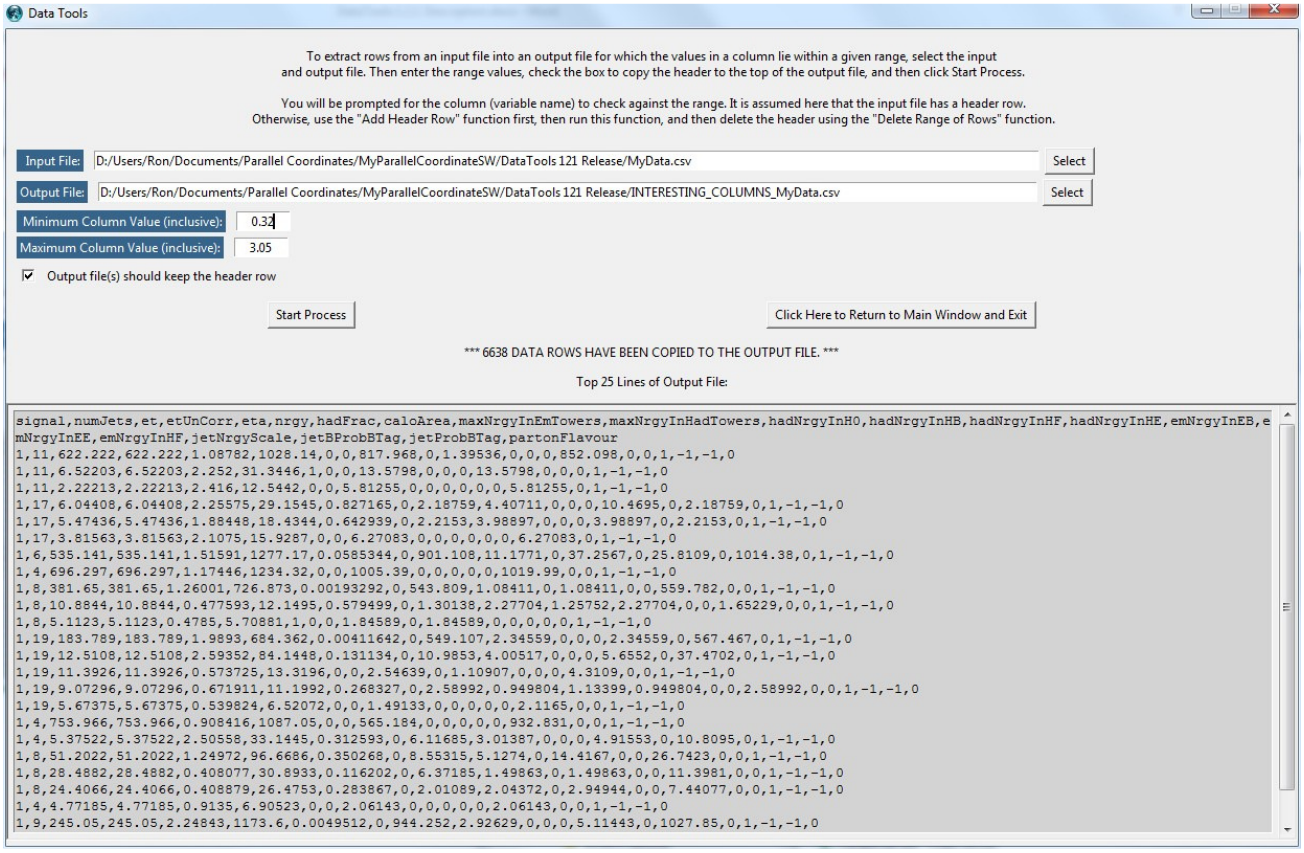

Figure 3. Extract Rows by Column Value Range window.

(The variable/column selection window appears after "Start Process" is clicked.)