Tablecruncher Documentation

The lightning-fast CSV editor for your Mac

tablecruncher.com

2019-01-26

Version: 0.9.7 beta

Homepage: https://tablecruncher.com/

Supported file formats

Tablecruncher is a CSV editor. While there is no offical definition for CSV files, this editor adheres to the de-facto standard that's given in RFC4180 [1]. Additionally Tablecruncher tries to be as liberal as possible when opening third-party CSV files.

Tablecruncher supports opening and storing CSV files with these separators: comma, semicolon, tab, colon(:), pipe(|).

Supported encodings for opening and saving:

- UTF-8
- UTF-16LE
- UTF-16BE
- Windows 1252
- Latin-1 (ISO-8859-1)

Support for more encodings, especially Latin-9 (ISO-8859-15), is planned for later versions.

Tablecruncher recognizes Unix style (LF = 0x0A or *n*), windows style (CRLF = 0x0D0A or *r**n*) and classic MacOS style (CR = 0x0D or *r*) line endings automatically.

1 https://tools.ietf.org/html/rfc4180

Opening a CSV file

When opening a CSV file, Tablecruncher guesses the encoding as well as the used separator of the given file. If Tablecruncher is not convinced that the guess is sufficiently correct, it opens an additional dialog and asks the user to provide the correct encoding as well as the used separator. If the file is roughly 50 MB or larger Tablecruncher always asks the user to provide the correct format, because guessing is a rather time consuming task.

There's an "Open with format …" function that always asks the user for the used file format. This is intended for cases when you think that Tablecruncher will guess the format incorrectly. If Tablecruncher opened a file with incorrect settings, just use "Reopen …" to re-open it while asking you for the correct import settings.

You can also open a file by dropping it onto the application symbol in your dock.

Storing a CSV file

An opened CSV file is stored using the format that's showing in the toolbar. To change this just click it and edit the settings. When you want to change the format of a CSV file, you'll open the file, click the setting in the toolbar and edit it as desired. Then just hit the "Save" button to store the CSV file.

Editing

To edit a cell just start typing. Typing TAB or ENTER finishes editing and stores the entered content in the cell, while ESC undos the previous change.

To add a new line within a cell, enter CTRL-J. Inserting a TAB character is done by CTRL-I.

Copy and paste, insert and deleting rows as well as columns should work as expected. Paste interprets the data stored in the clipboard as CSV and parses it before inserting. The command "Paste into Selection" pastes the content of the clipboard into all currently selected cells, in this case without any CSV parsing.

When copying cells the currently selected CSV definitions are used. These definitions can be shown by File > Info and changed by clicking the text field left of "Headers" that should read something like "UTF-8 …" and shows the currently used CSV definition.

Undo has an infinite undo buffer. Be careful with large files! Changes like deleting or adding a row or a column store a copy of the whole table in memory. You may run out of memory and your system may become rather slow. Editing single cells though doesn't copy the whole table and should not cause those problems.

Other functions

Find and replace functionality is provided using a single dialog box.

The "Info" dialog shows the filename (if given), the size of the table the CSV is representing and the format that is currently used.

When activating the "Header" option in the toolbar the first line of the CSV table is considered as a header row. The column names are then represented by the first row. To edit the header row, switch off "Header", edit the fields and then activate "Header" again.

In the "Edit" menu there are commands to move a column left or right.

"Sort …" in the "Find" menu lets you sort the rows of the table according to the content of the chosen column. You may also define the sorting order and the type of sorting (numerical or string). As a shortcut, right-clicking a colum header opens the sort window with the appropriate column prefilled.

Export JSON

File > Export JSON … exports the data in the current window as a JSON file. The JSON format depends on the kind of data you're using. When the "Header" option is activated, the data is stored as an array of objects, where the name of the header cells are the keys and the values are the cells from this column. Without an active header row, the exported JSON is just an array of arrays. Values are stored as integers, doubles or strings. If a cell with an otherwise valid integer or double number contains leading whitespace, the cell content is regarded as a number and the value is stored as such. When a cell contains trailing whitespace though, that cell is exported as a string.

Macros

Tablecruncher now offers a powerful macro language. You can use Javascript to access and modify the data in the CSV table.

Use *Macro > Execute Macro …* to access the new macro functionality. On the left there are the macros that are alreay defined. You can add a new macro by clicking the "+" icon below the macro list or delete a macro by clicking "-". The large editor on the right contains the macro source code. Here you define your Javascript code that is executed when you click on the "Execute" button.

These API functions are present to access the CSV data in the editor:

- `getInt(r,c)`: gets the content of the cell at the r-th row and the c-th column as an integer.
- `getFloat(r,c)`: gets the content of the cell at the r-th row and the c-th column as a float.
- `getString(r,c)`: gets the content of the cell at the r-th row and the c-th column as a string.
- `setCell(r,c,value)`: sets the content of the cell at the r-th row and the c-th column to the given "value".
- `println(val1, val2, ...)`: shows the contents of val1, val2 etc. in the macro window log area. A line break is added after every call to println.

All indices to the API functions are 0-based. So the top-most cell is called A1 in the interface, but it is addressed by r=0 and c=0. To address cell C2 for example, you have to use these parameters: r=1 and c=2.

The selected cells are provided to the Javascript source in the predefined variables ROWMIN, ROWMAX, COLMIN and COLMAX. A common task is to operate on all rows in a given colum. To do this select a colum and use the following Javascript loop to access all cells in the selected column:

```
for( r = ROWMIN; r<=ROWMAX; ++r ) {
        // do something using the provided API functions
}
```

The button "Insert Loop" inserts that loop code at the cursor position to make macro development even easier.

For example the following code normalizes the numeric values in the selected column so that the maximum value in that column is 1.

```
  // Normalize
  var max = getFloat(ROWMIN,COLMIN);
  for( r = ROWMIN+1; r<=ROWMAX; ++r) {
    if( getFloat(r,COLMIN) > max ) {
      max = getFloat(r,COLMIN);
    }
  }
  if( max > 0 ) {
    for( r = ROWMIN; r<=ROWMAX; ++r) {
      var elem = getFloat(r,COLMIN) / max;
      setCell(r,COLMIN,elem);
    }
  }
```