

---

# Tablecruncher Documentation

The lightning-fast CSV editor for your Mac

[tablecruncher.com](http://tablecruncher.com)



Version: 0.9.8 beta

Homepage: <https://tablecruncher.com/>

## Supported file formats

Tablecruncher is a CSV editor. While there is no official 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 (by default it reads "UTF-8|COMMA"). To change this just click this icon 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. You can also use "File > Set CSV Properties ..." to get to that property window.

As of beta version 0.9.8 you can also choose your desired quoting style:

- **Only when needed:** That's the current quoting strategy where only fields that are required by RFC 4180 are quoted.
- **On all fields:** Every field is quoted, no matter what the content looks like.
- **Around all strings:** Only fields that look like strings are quoted.

According to RFC 4180 only fields containing a line break, double quotes or the separator character are required to be quoted. Some CSV tools though expect all fields to be quoted, while other tools expect only non-numeric fields to be enclosed in quotes. With this option, you're able to create CSV files that should satisfy all the different expectations of those tools.

## Dealing with Escape characters

According to RFC 4180, there's no escape character needed. A quote within a cell is stored as two quote characters. This means that the quote character is effectively used as an escape character as well. Since 0.9.8, when opening files you can choose whether you want a quote or a backslash as an escape character. Keep in mind though that this choice is only valid for opening a file. When you store a file Tablecruncher always uses a quote character to escape a quote in a cell.

## 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. Pressing BACKSPACE clears the content of the currently selected cells.

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

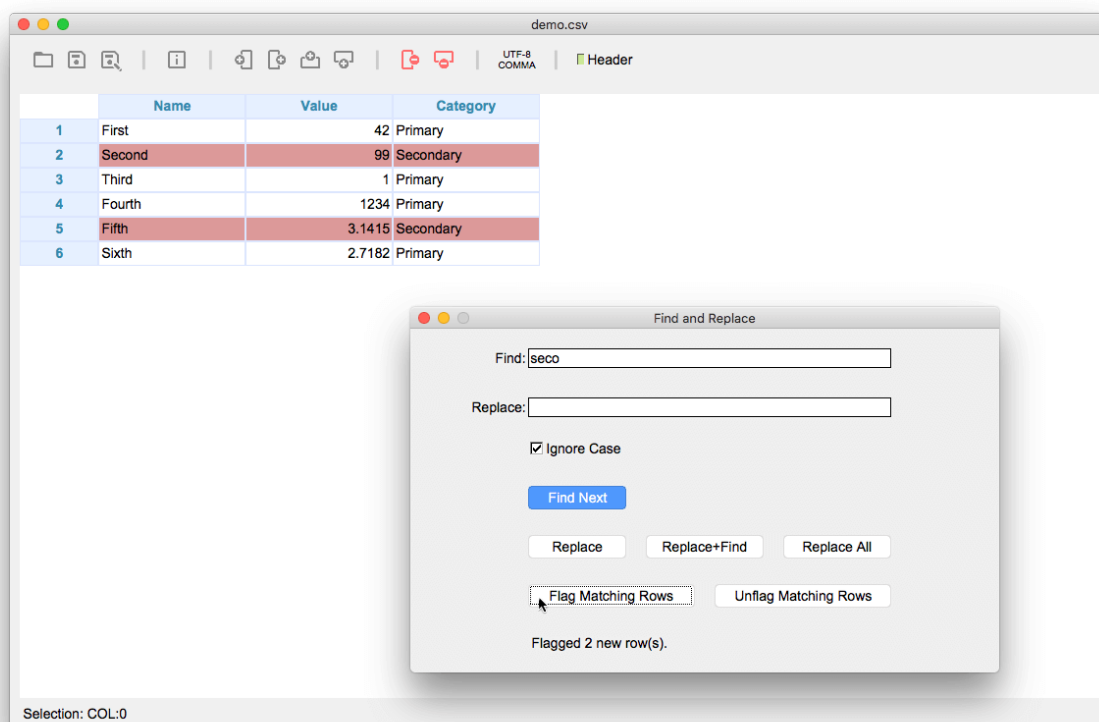
Copy and paste, insert and deleting rows as well as columns should work as expected. Paste interprets the data stored in the clipboard as CSV and parses it before inserting. The command "Paste into Selection" pastes the content of the clipboard into all currently selected cells, in this case without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

## Selecting and flagging cells

A selection are one or more adjacent cells that have been highlighted. To select multiple cells you can use the keyboard as well as the mouse. Click and drag with your mouse selects all cells from the cell you clicked to the cell you're pointing at when you release the mouse. If you want to select a really large selection of your table, it's easier to select a cell and shift-click another cell. Now all cells between those cells are selected. Pressing shift and one of the cursor keys also selects multiple cells. To select one complete column just click onto that column header. If you want multiple adjacent columns to be selected, click the first column header and drag your mouse over those other columns you want to be selected. The same works for rows accordingly.



*The improved Find and Replace dialog lets you easily flag rows by their content.*

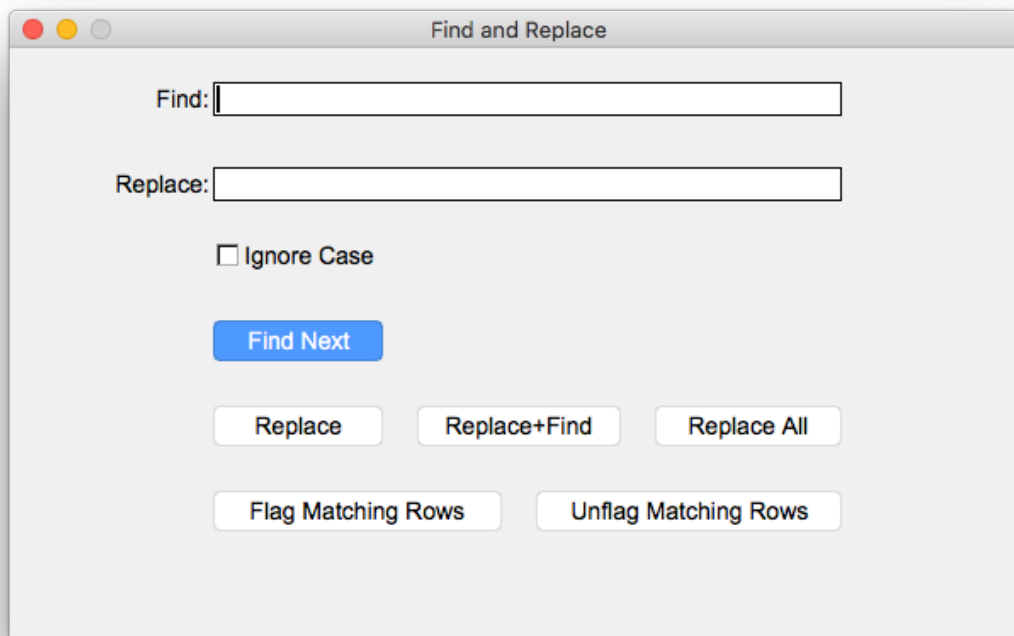
Since version 0.9.8-beta you can flag an arbitrary number of rows. Just click onto the row header while pressing the option (alt) key. Flagged rows are shown with a red background and need not be adjacent. This is especially helpful when working with Macros: You can flag rows based on all kind of conditions by your macro code. Then you may review those rows and for example delete them with the command "Data > Delete Flagged Row(s) ...". You can unflag a flagged row by option-clicking on the row header of a flagged row. Another very convenient way of flagging and unflagging rows is by using the Find & Replace dialog.

Using "View > Jump to Next Flagged Row ..." you can traverse all flagged rows in your table. The shortcut for this function is CMD-j. There's also a function to jump to the previous flagged row, just press CMD-SHIFT-j.

## Find and Replace

Find and replace functionality is provided using a single dialog box. If there are no cells selected, the complete table is searched. Clicking "Find Next" or pressing ENTER starts searching. If a cell is found it is highlighted with a green background. The next find starts from that last found position. When you

close the search dialog the last found position is reset.



"Replace" replaces the text in "Find" with the text in "Replace", but only at the cell where the last match occurred or at the selected cell. "Replace+Find" replaces the text in the current cell and starts a new search from that cell. "Replace All" replaces all occurrences of "Find" with "Replace" in the whole table or in the selection if an area has been selected.

"Flag Matching Rows" flags all rows if a cell in this row is successfully found. Searches the complete table or only within the selection when multiple cells have been selected.

If "Ignore Case" is selected all searching is done ignoring the case of the text to find.

## Other functions

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 "Data" 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 column header opens the sort window with the appropriate column prefilled. When using the shortcut, Tablecruncher preselects the sort type based on the column content.

## 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 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 already 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".
- `flagRow(r)`: the *r*-th row is flagged and can later be deleted by the user.
- `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 column. To do this select a column 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);  
    }  
}
```