Hugo Coto Flórez (hugo.coto@outlook.com)

Vicel Reference Manual

Contents

Abstract

Vicel is a free open source TUI spreadsheet editor. You can read, modify and save data stored in rows and columns. It aims to be an alternative to privative non gratis well known Microsoft program, for non professional usage (as it is quite simple, al least for now).

About development

For now it is being developed by Hugo Coto as a side project. He do that because for college reasons he has to use the Windows-only mouse-centered similar and he almost went ill. The plan for the future is to reach a stable version with all and no more than the useful and needed features to have an usable program for non professional usage. What I mean with no professional is that it's not planned to support any graphs neither economic formulas or such specific things. It's true that with builtins someone can adapt vicel to his own necessities.

Installation

This section would guide you to install vicel from source. Source is available in github. First, it's needed to clone the repo to your own machine.

```
git clone "https://github.com/hugoocoto/vicel"
cd vicel
```

Then, there are two options to install it.
1. Local installation: run make.
2. Global installation: run make install. This would move the executable to ~/.local/bin, make sure this route is in path.

After installation, it would be available. Note that local installation requires ./vicel while if installed globally it can be called just by name: vicel.

Open Vicel

As vicel is a TUI program you have to start it from the command line. If you installed it in ~/.LOCAL/ BIN then you can start it as a normal terminal tool:

```
vicel filename [options]
```

If the filename is omitted it opens an unnamed sheet. Note that this sheet can't be saved. Also, you can't open more than one file at the same time.

Options

The options supported are the following:

Command	Description
-m, --use-mouse	Enable mouse support
-D, --debug	Enable debug output
-c, --config-file	Set custom file path

For example, if you want to open the file ./SHEETS/TABLE.CSV with a config file in ./CONFIG/ VICEL.TOML, the command line should looks like that:

```
vicel sheets/table.csv -c config/vicel.toml
```

Moving around

As a vim enthusiast, movement is keyboard centered, and use the vim default HJKL. Every action can be prefixed with a number, so it would be executed that amount of times. The following table describes the basic movement.

Command	Description
h, l, j, k	Move cursor left, right, down, up
$	Go to last cell of the current row
^	Go to first cell of the current row
gg	Go to first cell of the current column
G	Go to last cell of the current column
g0	Same as ^ and gg

Write or edit

For write text in a cell, move the cursor there an press i. A text input box would be open at the cell. After write, press ENTER to save it. The data type would be automatically calculated by the program. Every number, with or without a fractional part separated by a dot would be interpreted as a NUMBER. If the text written starts with a equal sign it would be interpreted as a formula. Other formats would be set to TEXT.

Command	Description
i	insert/modify text
d	delete cell content
v	toggle cell selection

The valid types are described in the following table by it's formal representation.

Type	Formal
NUMBER	[0-9]+(“.”[0-9]+)?
FORMULA	“=” FORMULA BODY
TEXT	!NUMBER && !FORMULA

Formula

Formulas are expressions that evaluate to a valid value. They start with an equal sign. The function body have to contain a valid expression.

Type	Description	Example
Literal	Number, text, identifier or range	see below
Number	As cell type NUMBER	5987, 45.6
Text	Alphas or text surrounded by '	hello, '5.9'
Identifier	Cell reference by name as COLROW	A0, b5, ZZ98
Range	Cell range as ID:ID	A0:A2, A7:C8
Arithmetic operators	Evaluate arithmetic expressions	+, -, /, *, ^
Comparison operators	Compare two expressions	>, <, >=, <=, ==, !=, !
functions	Reserved names that convert some input in some output, with the form NAME(ARGS,...)	sum(A0,A1)

Todo: expand formula reference

Advanced write: write and move

There is a builtin feature to automatically move before insert text. It is useful if you need to input a big amount of data in a given direction. The idea is to prefix the following commands with a number, to do it for a given amount of times.

Command	Description
gih, gij, gik, gil	insert text and move in the given direction

Modify sheet structure

There is some commands to add/delete rows and columns. Note that formula identifier would not change on row/col insertion/deletion.

Command	Description
g#	Add row/column: see below
gd#	Delete row/column: see below
gj	Add a new row after the cursor
gl	Add a new column after the cursor
gk	Add a new row before the cursor
gh	Add a new column before the cursor
gJ	Add a new row at the end
gL	Add a new column at the end
gK	Add a new row at the start

Command	Description
gH	Add a new column at the start
gdj	Delete row and move up
gdl	Delete column and move right
gdk	Delete row and move down
gdh	Delete column and move left

Expand cells

There is a feature to get the cell value given the previous one and a direction. Numbers add 1 and formula identifiers recalculate depending on the direction. They can avoid modification if $ is used before identifier letter (for freeze column) or number (for freeze row). The mappings to do this are described in the following table.

Command	Description
J, K, H, L	Expand current cell down, up, left, right (and move)

Copy - Paste

As a vim user, you would want to copy-paste things around. Unfortunately, it's only possible to copy a single cell value and paste it in a single cell. Note that deletion also copy the content of the cell, it would sound natural for vim users.

Command	Description
p	paste
y	yank (copy)

Other commands

There are another useful commands, described below.

Command	Description
q	Save and quit
w	Write (save)
r	Re-render the screen
Ctrl-c	Quit without save

Mouse support

Despite the early development idea was to create a fully mouseless experience, some users may find convenient to do some actions with their mouse. It can be enabled setting the option WINDOW.USE_MOUSE to TRUE.

This is an experimental feature. At the time of writing, the cursor follows the mouse and you can drag and drop cell values using left click (drop on press, drag on release).

Configuration

vicel.toml

Because of the TOML parser it doesn't accept comments. This would be fixed in the future.

Color

Options in this table controls colors in all the editor.

```
[color]
ui = "49;30"                    # All ui text except ui_text_cell
ui_cell_text = "49;39;1"        # Cell text representation and previous message
ui_report = "41;39"             # Error/report message at the bottom right
cell = "49;39"                  # Cell color if not custom color applied
cell_over = "49;39;7;1"         # Cell color if cursor is over cell
cell_selected = "49;32"         # Cell color if selected
ln_over = "49;32;7;1"           # Row/col number/alpha if cursor is in this row/col
ln = "49;32"                    # Row/col number/alpha default color
sheet_ui = "49;39"              # UI elements inside sheet as separators
sheet_ui_over = "45;39;7;1"     # UI elements inside sheet if cursor is over they
sheet_ui_selected = "45;32"     # UI elements inside sheet if assigned cell is
selected
insert = "49;39"                # Color used when cell input text is being written
```

Window

Despite of the name, in this table lives all the options that are not colors. In the future it would migrate to another table.

```
[window]
num_col_width = 5               # Number column width
col_width = 14                  # Column width (min is cell_l_sep + cell_r_sep + 1)
row_width = 1                   # Other size is not supported
use_cell_color_for_sep = true   # Use cell color for separators instead of sheet_ui
cell_l_sep = " "                # Left separator
cell_r_sep = " "                # Right separator
save_time = 0                   # Time interval (in seconds) where save is call. 0
means no autosave.
use_mouse = false               # Enable mouse capturing
```

This is the ui customization, where you can modify how the editor looks like (Also in WINDOW).

```
# Top bar
status_l_stuff = "vicel | "     # Top Left bar text
status_filename = "filename: "  # Between status_l_stuff and filename
status_r_end = "github: hugoocoto/vicel" # Top right-align bar text

# Bottom bar
ui_celltext_l_sep = "cell text: " # Bottom Left bar text, before cell repr text
ui_celltext_m_sep = " ("        # Between cell text and cell type
ui_celltext_r_sep = ") "        # Before cell type, left-aligned
ui_status_bottom_end = ""       # Bottom right-align text
```