I mentioned [EtherCalc in a previous post](https://rud.is/b/2019/06/11/makeover-jumbalaya-beating-dumbbells-into-slopegraphs-whilst-orchestrating-ethercalc/) and managed to scrounge some time to put together a fledgling [{ethercalc} package](https://git.rud.is/hrbrmstr/ethercalc) (it’s also on GitLab, SourceHut, Bitbucket and GitUgh, just sub out the appropriate URL prefix).

I’m using a package-specific Docker image (there are a couple out there but I’m not supporting their use with the package as they have a [CORS](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS) configuration that make EtherCalc API wrangling problematic) for EtherCalc but I would highly recommend that you just use it via the npm module. To do that you need a working Node.js installation and I highly recommended also running a local Redis instance (it’s super lightweight). Linux folks can use their fav package manager for that and macOS folks can use homebrew. Folks on the legacy Windows operating system can visit:

* Download Node.js,
* Get npm, and
* Running Redis on Windows 10

to get EtherCalc going.

I also recommend running EtherCalc and Redis together for performance reasons. EtherCalc will maintain a persistent store for your spreadsheets (they call them “rooms” since EtherCalc supports collaborative editing) with or without Redis, but using Redis makes all EtherCalc ops much, much faster.

Once you have Redis running (on localhost, which is the default) and Node.js + npm installed, you can do the following to install EtherCalc:

$ npm install -g ethercalc # may require `sudo` on some macOS or \*nix systems

The -g tells npm to install the module globally and will work to ensure the ethercalc executable is on your PATH. Like many things one can install from Node.js or, even Python, you may see a cadre of “warnings” and possibly even some “errors”. If you execute the following and see similar messages:

$ ethercalc --host=localhost ## IMPORTANT TO USE --host=localhost

Please connect to: http://localhost:8000/

Starting backend using webworker-threads

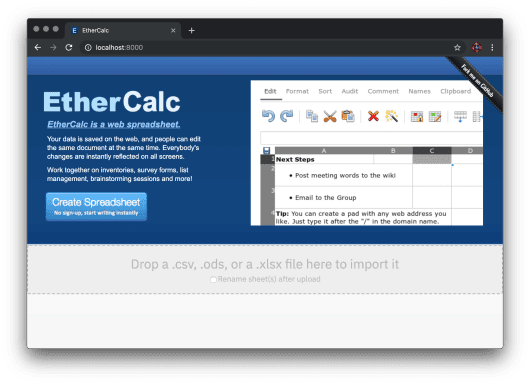
Falling back to vm.CreateContext backend

Express server listening on port 8000 in development mode

Zappa 0.5.0 "You can't do that on stage anymore" orchestrating the show

Connected to Redis Server: localhost:6379

and then go to the URL it gives you and you see something like this:

[](https://i2.wp.com/rud.is/b/wp-content/uploads/2019/06/ethercalc-splash.png?ssl=1)

then you’re all set to continue.

**A [Very] Brief EtherCalc Introduction**

EtherCalc has a wiki. As such, please hit that to get more info on EtherCalc.

**EtherCalc wiki**

EtherCalc is the spreadsheet equivalent of a wiki document, without logins and without editing locks. People can collaborate on a spreadsheet-like page simultaneously. It's somewhat a google sheet alternative, open source style.

**Use Cases**

When or why would you use EtherCalc? Here are a few scenarios.

* Scheduling a meeting without asking via e-mails nor filling one-time forms: Every participant can see the most up-to-date availability of other members. One can update his/her schedule any number of times. The number of available participants in each time slot can be easily seen.
* Car pooling arrangement: Who will ride in whose car? How many seats are still available in each car?
* Students mutual-grading: What does each student think of each other's exercises or projects?
* A good communication tool between the Direction of Programs and the Studio in a radio station : the director writes the roadmap in a calc sheet and the technician at the studio can easily follow along and receive the latest modifications without any interaction needed (no required clicks, no required refresh).
* ...

**Syntax**

As a spreadsheet, ethercalc accepts the following in a cell:

* mathematical formulae or other functions beginning with the equal sign =, e.g. =sqrt(3)/2, =if(1+1<0, 'surprise!', 'ok'), ... See [OpenFormula](https://en.wikipedia.org/wiki/OpenFormula) and the SocialCalc.Formula.FunctionList variable in the source code for the complete list of formulas it supports. Certain functions (such as DAVERAGE, DCOUNT, DMAX, DSUM, COUNTIF, SUMIF, ...) require a criteria parameter. The criteria parameter can be either a number (possibly written as a string prefixed by one of the comparison operators <, <=, =, >=, >, <>) or a textual string. A textual string criteria containing either \* or ? will be interpreted as an excel wildcard string or as a javascript regular expression.
* date of the form (by default) mm/dd/yyyy where yyyy>=1900, e.g. 12/25/2016
* texts beginning with the single quote character ', e.g. 'hello world

For text cells, ethercalc supports the following formatting directives in its "default" formatting option:

* bold: the point is \*collaboration\*
* italic: the point is \_collaboration\_
* strikethrough: can I -haz- cheezburger
* teletype: it reads ``SOS`` (backquote character)
* horizontal line: -- (two or more dashes at the very beginning of the cell's text)
* header text of various levels: ^^Section 4 (one or more caret characters at the very beginning of the cell's text)
* hyperlink: Leonard Nimoy as "Spock"<https://en.wikipedia.org/wiki/Spock>

For even richer formatting options such as color text/background, you can select a range of cells, switch to the format tab, and change format:text from "default" to "html". Then you can type html syntax such as <span style="color:red; background:yellow">hello</span> in your cells. See e.g.: <http://www.w3schools.com/html/html_formatting.asp>

**Tips and tricks**

**Use a cell value from another sheet.**

*Example, I have two sheets named sheet1 and sheet2 and I want to display cell A1's value from sheet1 in the sheet2's cell B3.*

1. Go to cell B3 in sheet2
2. type this formula:="sheet1"!A1

**Color a cell depending of its value (aka Conditional Formating) .**

*Example, I want a cell B2 to turn green if its value is foo; or else to be red.*

1. Go to cell B2
2. In the menu/format/format/text choose "HTML" and save for this cell
3. Then type this formula: =if(A1="foo","<span style=""background-color:rgb(81,184,72);color:rgb(81,184,72)"">\_\_\_\_\_\_\_</span>","<span style=""background-color:rgb(226,86,43);color:rgb(226,86,43)"">\_\_\_\_\_\_\_</span>")

So how does it work? Your IF function checks A1's value and gives you two possible output each of them being a simple text string made of underscores. The only subtility here is that we specify both pen color and background color to be the same (either green or red) so as it looks like the cell is filled either in red or in green. This is a workaround since there is no current straitforward function to set an empty cell's background color.

**Create a form**

Say you have a calc named foo, which URL is <https://calc.example.org/foo>.

You can create a form by adding /appeditor to your URL (which become <https://calc.example.org/foo/appeditor>).

You can add '/form' to show the form - e.g. <https://calc.example.org/foo/form>

You screen will be split in three parts. To the left, the form as it will be rendered, to the right, your calc, which will be used to create the form, and at the bottom, the results from the form.

To create an input field, go in a cell and write =TEXTBOX("some text"). Note that you can't use =TEXTBOX(""), you have to put the quotes, but the content can be blank.

Have a look at the other form possibilities by clicking on the "Fx" button on the spreadsheet, and selecting "Button & Input" section.

To create a submit button, go in a cell and write =submit("send") (you can put any text you want).

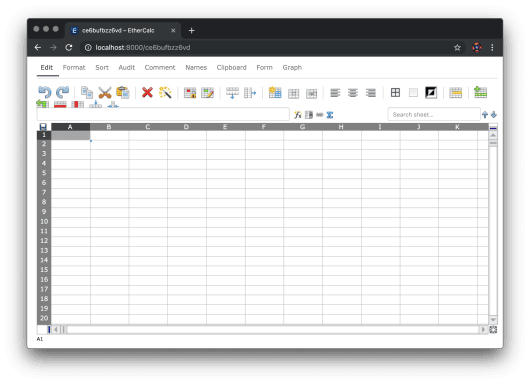
To get the form link to share, click on the "Form" tab of your calc, then on the "Live form" button. It will redirect you to your form: you only have to share that page's URL.

To get the results of your form, add \_formdata/view at the end of your calc's URL (which become <https://calc.example.org/foo_formdata/view>).

**Warning!** It's easy to get the results page's URL or the original calc's URL from the form URL. Don't use it to get private datas!

**Warning! (part 2)** There's currently [an issue](https://github.com/audreyt/ethercalc/issues/383) with forms on Ethercalc: the input fields lose focus after each keystroke.

For now, if you hit that big, blue “Create Spreadsheet” button, you’ll see something pretty familiar if you’ve used Google Sheets, Excel, LibreOffice Calc (etc):

[](https://i1.wp.com/rud.is/b/wp-content/uploads/2019/06/ethercalc-create.png?ssl=1)

If you start ethercalc without the --host=localhost it listens on all network interfaces, so other folks on your network can also use it as a local “cloud” spreadsheet app, but also *edit* with you, just like Google Sheets.

I recommend playing around a bit in EtherCalc before continuing just to see that it is, indeed, a spreadsheet app like all the others you are familiar with, except it has a robust API that we can orchestrate from within R, now.

**Working with {ethercalc}**

You can install {ethercalc} from the aforelinked source or via:

install.packages("ethercalc", repos = "https://cinc.rud.is")

where you’ll get a binary install for Windows and macOS (binary builds are for R 3.5.x but should also work for 3.6.x installs).

If you don’t want to drop to a command line interface to start EtherCalc you can use ec\_start() to run one that will *only be live during your R session*.

Once you have EtherCalc running you’ll need to put the URL into an ETHERCALC\_HOST environment variable. I recommend adding the following to ~/.Renviron and restarting your R session:

ETHERCALC\_HOST=http://localhost:8000

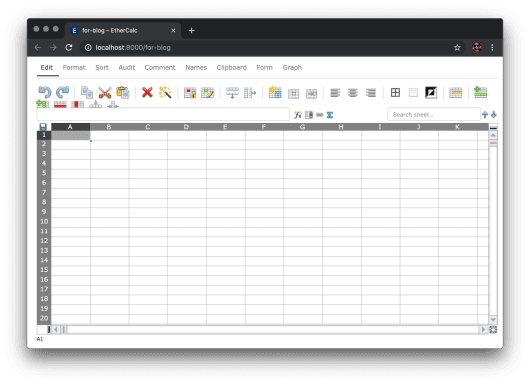
(You’ll get an interactive prompt to provide this if you don’t have the environment variable setup.)

You can verify R can talk to your EtherCalc instance by executing ec\_running() and reading the message or examining the (invisible) return value. Post a comment or file an issue (on your preferred social coding site) if you *really* think you’ve done *everything* right and still aren’t up and running by this point.

The use-case I setup in the previous blog post was to perform light data entry since scraping was both prohibited and would have taken more time given how the visualization was made. To start a new spreadsheet (remember, EtherCalc folks call these “rooms”), just do:

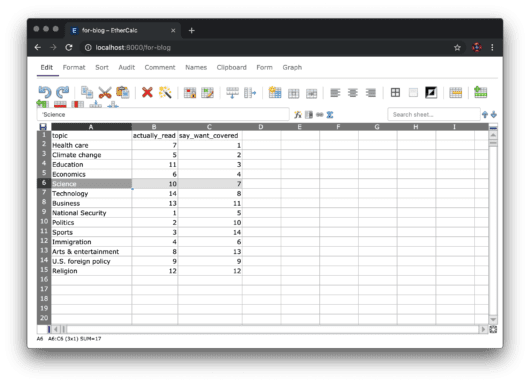
ec\_new("for-blog")

And you should see this appear in your default web browser:

[](https://i1.wp.com/rud.is/b/wp-content/uploads/2019/06/ethercalc-new.png?ssl=1)

You can do ec\_list() to see the names of all “saved” spreadsheets (ec\_delete() can remove them, too).

We’ll type in the values from the previous post:

[](https://i2.wp.com/rud.is/b/wp-content/uploads/2019/06/ethercalc-entry.png?ssl=1)

Now, to retrieve those values, we can do:

ec\_read("for-blog", col\_types="cii")

## # A tibble: 14 x 3

## topic actually\_read say\_want\_covered

##

## 1 Health care 7 1

## 2 Climate change 5 2

## 3 Education 11 3

## 4 Economics 6 4

## 5 Science 10 7

## 6 Technology 14 8

## 7 Business 13 11

## 8 National Security 1 5

## 9 Politics 2 10

## 10 Sports 3 14

## 11 Immigration 4 6

## 12 Arts & entertainment 8 13

## 13 U.S. foreign policy 9 9

## 14 Religion 12 12

That function takes any (relevant to this package use-case) parameter that readr::read\_csv() takes (since it uses that under the hood to parse the object that comes back from the API call). If someone adds or modifies any values you can just call ec\_read() again to retrieve them.

The ec\_export() function lets you download the contents of the spreadsheet (“room”) to a local:

* CSV
* JSON
* HTML
* Markdown
* Excel

file (and it also returns the raw data directly to the R session). So you can do something like:

cat(rawToChar(ec\_export("for-blog", "md", "~/Data/survey.md")))

## |topic|actually\_read|say\_want\_covered|

## | ---- | ---- | ---- |

## |Health care|7|1|

## |Climate change|5|2|

## |Education|11|3|

## |Economics|6|4|

## |Science|10|7|

## |Technology|14|8|

## |Business|13|11|

## |National Security|1|5|

## |Politics|2|10|

## |Sports|3|14|

## |Immigration|4|6|

## |Arts & entertainment|8|13|

## |U.S. foreign policy|9|9|

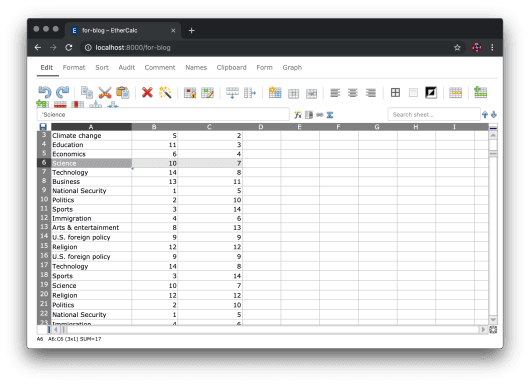
## |Religion|12|12|

You can also *append* to a spreadsheet right from R. We’ll sort that data frame (to prove the append is working and I’m not fibbing) and append it to the existing sheet (this is a toy example, but imagine appending to an always-running EtherCalc instance as a data logger, which folks actually do IRL):

ec\_read("for-blog", col\_types="cii") %>%

dplyr::arrange(desc(topic)) %>%

ec\_append("for-blog")

[](https://i2.wp.com/rud.is/b/wp-content/uploads/2019/06/ethercalc-append.png?ssl=1)

Note that you can open up EtherCalc to any existing spreadsheets (“rooms”) via ec\_view() as well.

**FIN**

It’s worth noting that EtherCalc *appears* to have a limit of around 500,000 “cells” per spreadsheet (“room”). I mention that since if you try to, say, ec\_edit(ggplot2movies::movies, "movies") you would have very likely crashed the running EtherCalc instance if I did not code in some guide rails into that function and the ec\_append() function to stop you from doing that. It’s sane limit IMO an Google Sheets does something similar (per-tab) for the similar reasons (and both limits are one reason I’m still against using a browser for “everything” given the limitations of javascript wrangling of DOM elements).

If you’re doing work on large-ish data, spreadsheets in general aren’t the best tools.

And, while you should avoid hand-wrangling data at all costs, ec\_edit() is a much faster and feature-rich alternative to R’s edit() function on most systems.

I’ve shown off *most* of the current functionality of the {ethercalc} package in this post. One function I’ve left out is ec\_cmd() which lets you completely orchestrate *all* EtherCalc operations. It’s powerful enough, and the EtherCalc command structure is gnarly enough, that we’ll have to cover it in a separate post. Also, stay tune for the aforementioned package-specific EtherCalc Docker image.

Kick the tyres, contribute issues and/or PRs as moved (and on your preferred social coding site) and see if both EtherCalc and {ethercalc} might work for you in place of or along with Excel and/or Google Sheets.