

Generating documents with knitr and Clojure

This html document is rendered by R `knitr` package with embedded Clojure code. Yes, it's possible. The renderer is configured to use `nRepl` client: `rep`.

What is knitr in short

Knitr is R package which generates really variety documents out of markdown file with embedded code.

First run

First let's define data.

```
(def data {:a [1 2 3]
           :b [3 4 5]})
```

```
#'user/data
```

Code was executed, `data` is defined and we can run another chunk.

```
(keys data)
```

```
(:a :b)
```

And another one.

```
(->> data
  vals
  (apply concat)
  (reduce +))
```

```
18
```

Generate image

```
(require '[clojure2d.core :refer [save]]
         '[clojure2d.color :as c]
         '[clojure2d.extra.utils :as u])
```

```
nil
```

```
(def img (-> :cubelix
             (c/gradient)
             (u/gradient->image true)
             (save "gradient.png")))
```

```
saving: gradient.png...
```

```
...done!
```

```
#'user/img
```

How to setup

I'm using Emacs with CIDER here.

- Clojure

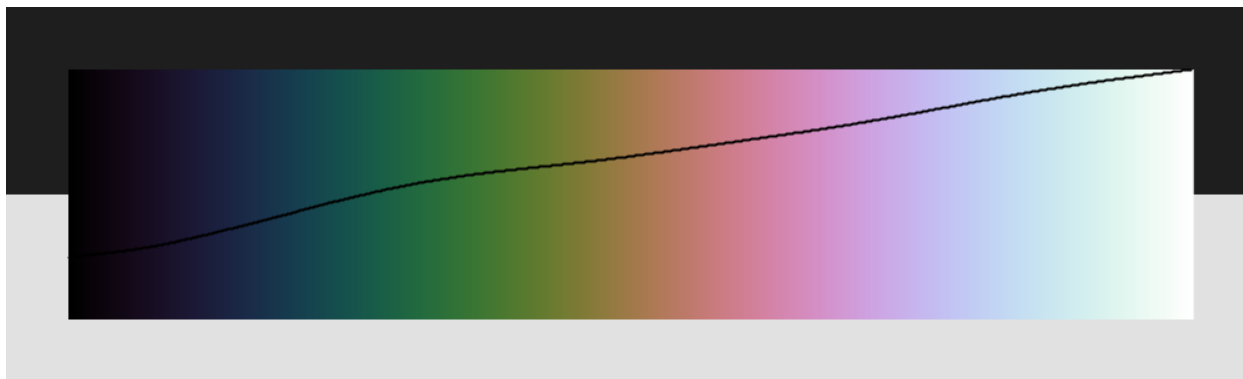


Figure 1: Generated gradient with luma

- Download and install `rep`
- Be able to run `nRepl`
- R
 - Install R with `knitr` package (and all needed deps, like `pandoc`)
- Emacs
 - Install ESS, `poly-R` package which enables REPL inside Markdown file.

Run `nRepl`, create `.Rmd` file and add below chunk at the beginning of it. As you can see, there is a place to define `nrepl_port`. Find your port and change this value. I haven't been able to find an easy way to setup it automatically (yet).

```
```{r setup, include=FALSE}
nrepl_port <- "53247"
library(knitr)
knitr_one_string <- knitr::one_string
nrepl_cmd <- "rep"
opts_chunk$set(comment=NA, highlight=TRUE)
knit_engines$set(clojure = function(options) {
 code <- paste("-", nrepl_port, shQuote(knitr_one_string(options$code)))
 out <- if (options$eval) {
 if (options$message) message('running: ', nrepl_cmd, ' ', code)
 tryCatch(
 system2(nrepl_cmd, code, stdout = TRUE, stderr = TRUE, env = options$engine.env),
 error = function(e) {
 if (!options$error) stop(e)
 paste('Error in running command', nrepl_cmd)
 }
)
 } else ''
 if (!options$error && !is.null(attr(out, 'status'))) stop(knitr_one_string(out))
 engine_output(options, options$code, out)}
```
```

When it's done you can generate documents (html, pdf, whatever) within ESS or from external R session.

```
render("README.Rmd", "all")
```

What's odd

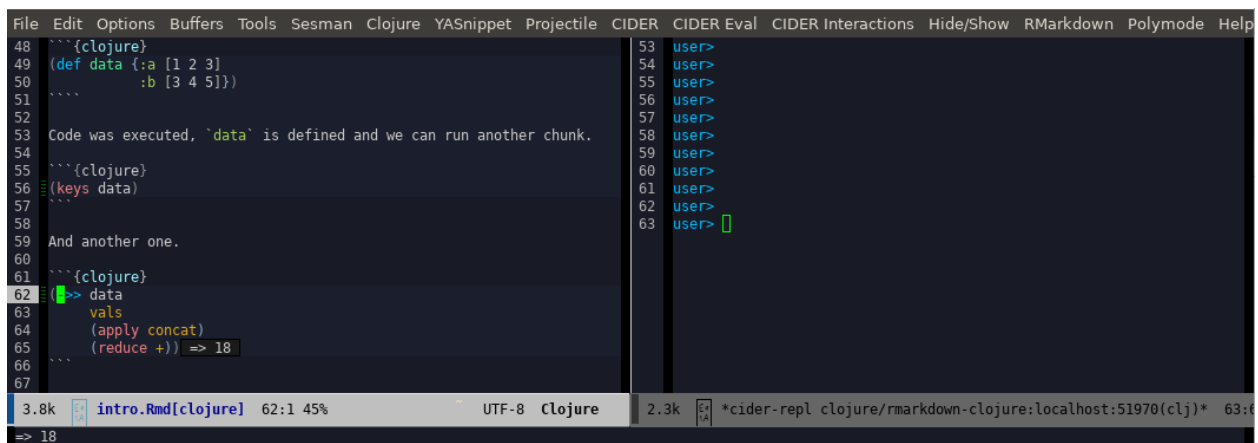
There are couple of problems:

- manual renderer setup
- manual port setup
- no pretty printing results by default

RMarkdown reference

<https://bookdown.org/yihui/rmarkdown/>

Emacs view



The screenshot shows the Emacs editor interface with a dark theme. The main window displays RMarkdown code in a Clojure chunk. The code defines a variable 'data' and then uses it in a calculation. A CIDER REPL window is open on the right, showing the execution of the code. The status bar at the bottom indicates the file is 'intro.Rmd[clojure]' and the current buffer is '*cider-repl clojure/rmarkdown-clojure:localhost:51970(clj)*'.

```
File Edit Options Buffers Tools Sesman Clojure YASnippet Projectile CIDER CIDER Eval CIDER Interactions Hide/Show RMarkdown Polymode Help
48 ```{clojure}
49 (def data {:a [1 2 3]
50           :b [3 4 5]})
51 ....
52
53 Code was executed, `data` is defined and we can run another chunk.
54
55 ```{clojure}
56 (keys data)
57 ....
58
59 And another one.
60
61 ```{clojure}
62 (c-> data
63   vals
64   (apply concat)
65   (reduce +)) => 18
66 ....
67
```

3.8k intro.Rmd[clojure] 62:1 45% UTF-8 Clojure 2.3k *cider-repl clojure/rmarkdown-clojure:localhost:51970(clj)* 63:1

Figure 2: Emacs in action