Knitting Clojure snippets

GenerateMe

2020-05-17

Generating documents with `knitr` and Clojure

This 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.

Let's run something

First let's define `data`.

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

```
#'user/data
{:a [1 2 3], :b [3 4 5]}
```

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

```
(keys data)
```

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

And another one (everything is kept in `user` namespace).

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

```
saving: gradient.png...
...done!
#'user/img
```

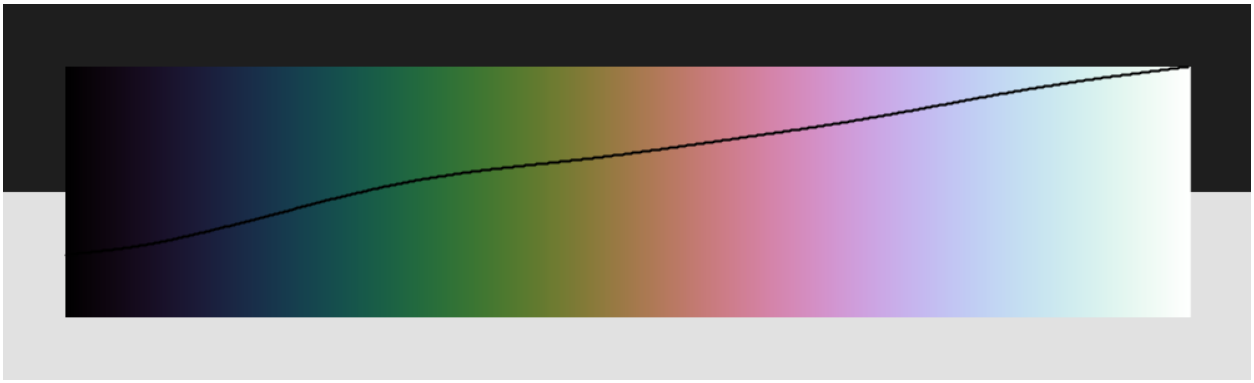

Figure 1: Generated gradient with luma

Generate markdown

```
(println "
test/data/stocks.csv [5 3]:

| symbol |       date | price |
|--------+------------+-------|
|   MSFT | 2000-01-01 | 39.81 |
|   MSFT | 2000-02-01 | 36.35 |
|   MSFT | 2000-03-01 | 43.22 |
|   MSFT | 2000-04-01 | 28.37 |
|   MSFT | 2000-05-01 | 25.45 |
")
```

test/data/stocks.csv [5 3]:

symbol	date	price
MSFT	2000-01-01	39.81
MSFT	2000-02-01	36.35
MSFT	2000-03-01	43.22
MSFT	2000-04-01	28.37
MSFT	2000-05-01	25.45

How to setup

I'm using Emacs with CIDER here.

- Clojure
- Download and install `rep`
  - Be able to run `nRepl`
- R
  - Install R with `knitr` and `rmarkdown` packages (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}
find_nrepl_port_up <- function() {
    wd <- getwd()
    while(wd != dirname(wd)) {
        f <- paste0(wd,"/.nrepl-port")
        if(file.exists(f)) return(paste0("@",f))
        wd <- dirname(wd)
        f <- NULL
    }
}
port_file <- find_nrepl_port_up()
if(is.null(port_file)) stop("nREPL port not found")
library(knitr)
knitr_one_string <- knitr:::one_string
nrepl_cmd  <- "rep"
opts_chunk$set(comment=NA, highlight=TRUE)
knit_engines$set(clojure = function(options) {
    rep_params <- if(options$results=="asis") {
                      "--print 'out,1,%{out}' --print 'value,1,' -p"
                  } else {
                      "-p"
                  }
    code <- paste(rep_params, port_file, shQuote(knitr_one_string(options$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.

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

Emacs view

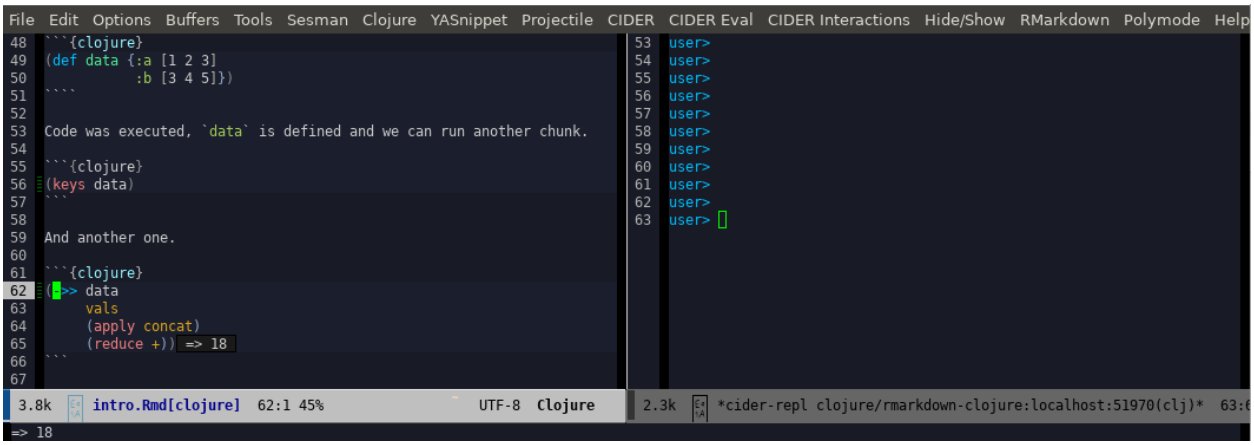

Figure 2: Emacs in action

Rendered documents

- HTML
- PDF
- WORD

What's odd

There are couple of problems:

- manual renderer setup
- no pretty printing results by default

RMarkdown reference

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