# 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](https://github.com/eraserhd/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



Generated gradient with luma

## How to setup

I’m using Emacs with CIDER here.

* Clojure
  + Download and install [rep](https://github.com/eraserhd/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.

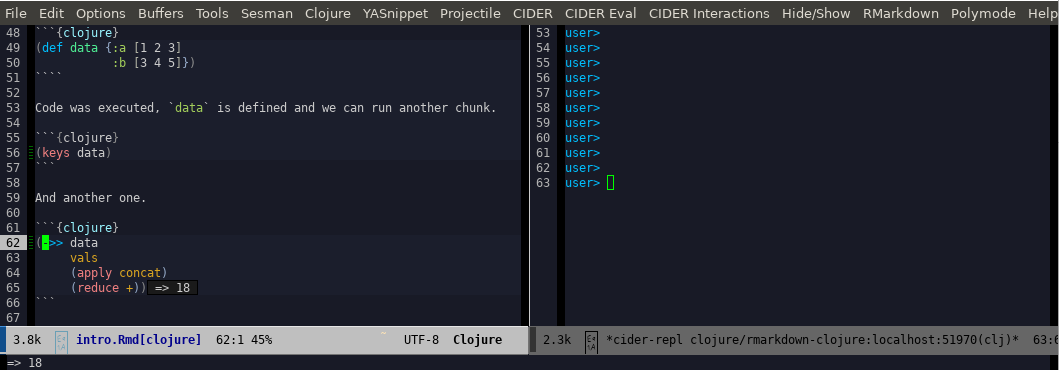
Run nRepl, create .Rmd file and add below chunk at the beginning of it. As you can see, there is a place to define nrepl\_port. Find your port and change this value. I haven’t been able to find an easy way to setup it automatically (yet).

```{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) {  
 code <- paste("-p", 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.

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

## Emacs view



Emacs in action

## Rendered documents

* [HTML](https://genmeblog.github.io/rmarkdown-clojure/README.html)
* [PDF](https://github.com/genmeblog/rmarkdown-clojure/blob/master/README.pdf)
* [WORD](https://github.com/genmeblog/rmarkdown-clojure/blob/master/README.docx)

## What’s odd

There are couple of problems:

* manual renderer setup
* no pretty printing results by default

## RMarkdown reference

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