Language engine for including data in Rmarkdown

David M. Kaplan

3/27/2020

Overview

This document contains a proof of concept for adding data engines to Rmarkdown that would allow placing data directly inside Rmarkdown documents to create completely self-contained Rmarkdown documents. The implementation is based on the idea that `data` chunks will contain the contents of the data files, potentially encoded as text using some encoding method.

Though many types of data chunks are imaginable, I have currently implemented two: `csv` and `RDS`. For the `RDS` case, as the file format is binary, it must be encoded as text to be included in a chunk. Two different encodings are currently possible: `base64` and `GPG`.

The implementation is quite simple, but it works well. The model could quickly be extended to other

Implementation of language engines

```r
data_decode = function(code,encoding,options,file=tempfile()) {
  switch(
    encoding,
    asis = writeLines(code,file),
    base64 = writeBin(base64enc::base64decode(code),file),
    gpg = {
      tf = tempfile()
      writeLines(code,tf)
      on.exit(file.remove(tf))
      file.copy(tf,"temp_file.gpg")
      writeBin(gpg::gpg_decrypt(tf,as_text=FALSE),file)
    },
    stop("Uknown encoding: ",encoding)
  )

  return(file)
}

data_encode = function(file,encoding,options=list(base64.linewidth=64),output=NULL) {
  code = switch(
    encoding,
    asis = readLines(file),
    base64 = base64enc::base64encode(file,linewidth=options$base64.linewidth),
    gpg = {
      if (is.null(options$gpg.receiver))
```

```r
        stop("Missing GPG receiver. See ?gpg::gpg_encrypt for details.")
      gpg::gpg_encrypt(file,options$gpg.receiver,options$gpg.signer)
    },
    stop("Uknown encoding: ",encoding)
  )

  if(is.null(output)) {
    cat(code,sep="\n")
  } else {
    writeLines(code,output)
  }

  invisible(code)
}

eng_data = function(options) {
  vn = options$output.var
  if (is.null(vn))
    stop("output.var must be supplied in data chunks.")

  format = options$format
  if (is.null(format))
    format = 'csv'

  encoding = options$encoding
  if (is.null(encoding)) {
    encoding = switch(
      format,
      csv = 'asis',
      RDS = 'base64',
      'asis'
    )
  }

  encoding.ops = options$encoding.ops
  if (is.null(encoding.ops))
    encoding.ops = list()
  if (!is.list(encoding.ops))
    stop("encoding.ops should be a list. Got object of class ",class(encoding.ops)[1])

  format.ops = options$format.ops
  if (is.null(format.ops))
    format.ops = list()
  if (!is.list(format.ops))
    stop("format.ops should be a list. Got object of class ",class(format.ops)[1])

  fn = data_decode(options$code,encoding,options=encoding.ops)
  on.exit(file.remove(fn))

  #save(options,format,encoding,format.ops,encoding.ops,file="debug.RData")
  if (is.character(format)) {
    format = switch(
      format,
      csv = read.csv,
      RDS = readRDS,
      get(format) # Attempt format as function name
    )
  }
  data = do.call(format,c(file=fn,format.ops))

  assign(vn, data, envir = knitr::knit_global())
  knitr::engine_output(options,options$code,'')
}

knitr::knit_engines$set(data=eng_data)
```

Test of csv chunk

```
id,res
1,a
2,b
3,c
```

```r
t1
```

id	res
1	a
2	b
3	c

Test of RDS chunk

H4sIAAAAAAAAA4vgYmBgYGZgZgNiViCTgTUOxE3XgoGBSRjI
YQLid1CakYGFgRNI8yXn5xYkJpfEZ+aVFKcWosmyJCUWpOLF
eMHiEPofSCfIKgcVBjCw/4BKQ9UIIJnFnJiUDDQR2XjWvMTc
1GKoOiaoIGMijJEEMgUAP3IVTdMAAAA=

```r
names(t2)
```

```
## [1] "a" "b"
```

```r
t2
```

```
## $a
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $b
## [1] "abc"
```

Test of GPG encrypted RDS chunk

```r
names(t3)
```

```
## [1] "a" "b"
```

```
t3
```

```
## $a
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $b
## [1] "abc"
```

Encoding a binary file for incorporation in Rmarkdown

```
saveRDS(t2,"test.RDS")
```

base64

```
data_encode("test.RDS","base64")
```

```
## H4sIAAAAAAAAA4vgYmBgYGZgZgNiViCTgTUOxE3XgoGBSRjIYQLid1CakYGFgRNI
## 8yXn5xYkJpfEZ+aVFKcWosmyJCUWpOLFeMHiEPofSCfIKgcVBjCw/4BKQ9UIIJnF
## nJiUDDQR2XjWvMTc1GKoOiaoIGMijJEEMgUAP3IVTdMAAAA=
```

GPG

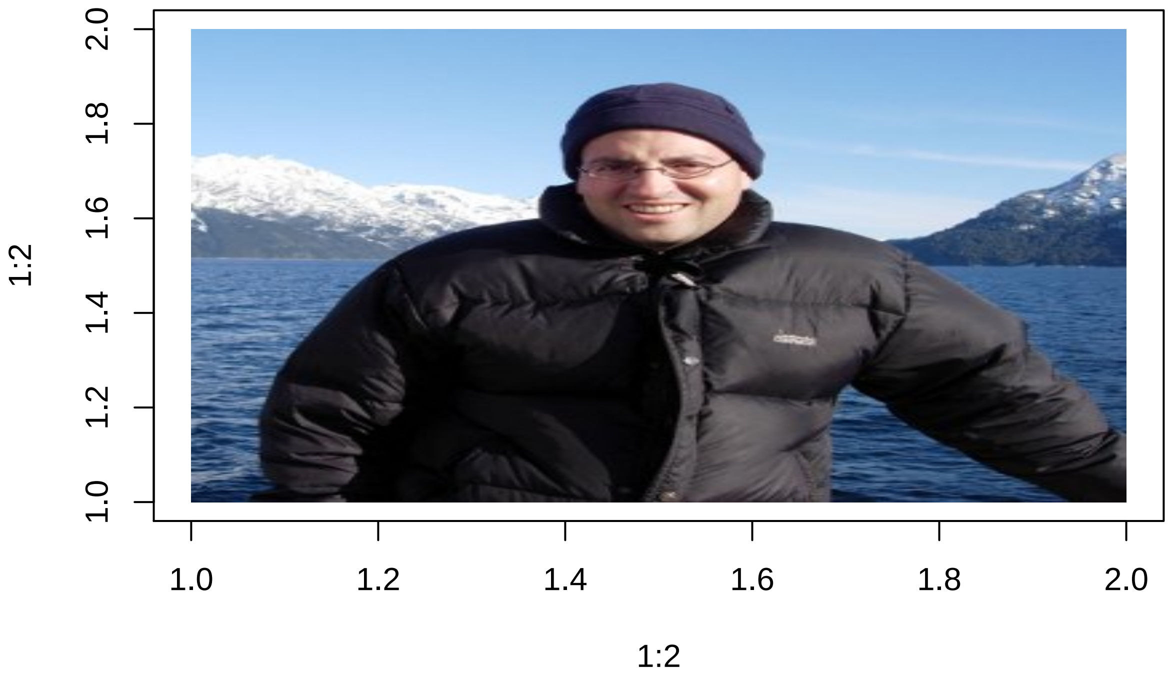
```
data_encode("test.RDS","gpg",options=list(gpg.receiver="8D8AFF6338C465E3"))
```

```
## -----BEGIN PGP MESSAGE-----
## Version: GnuPG v2
##
## hQEMA+gixT2HKBy2AQf/eFpJ3C4pcSor+AdYf4DqXdYvsLxPs4BcuT9jC7ZOlyak
## 2OK6c5zI6jnUnitYjvURa57RW4VNFkpU6jkbWlp+bqtQxbnJUE3gf1vKIupLCb2O
## cRjtc9Pbv9oWI9L+mTTP4yvfqfrDO4pmqB4CuYRm0xP6f9p14mS8FCOoR2fKtOfV
## Ah+EKi2Sw7YZfOnfkaEafgpu88jN3R6n+bC+6/7mZ5Ay5sY9r4M65Kp/QsOuQ++E
## OVpj8TyTsNE+bicyasyWTOgguJodBHcvTDDamnimkcTb9tUoplxTOY3tjvZMxEty
## o9ELW4imaJRKJpFLGr1djkNxFm/q+ZGxR9xkSjHhvtK/ATD4l2MRQ1FX3MFWG5iT
## 4Tvod/629JqaSAw/2ROO9TobRky8iIIsHc7lJSvDAeVWvHNYFgt12IteQHGgYT5x
## Z7FOvBjPvmGB6+U6haI7Nd6bNoGceKeOu1hlqBRPgXjHu8p3SQGlsfLQFO/LpSak
## biUxJIA2POpJggB7oeYbtOdVj+UEYp8X2mq1dQcFXZD6QvDTsQBKqayT6Dkf5bCe
## 7V4oYMYvmkI/OhgaP+foSC7PZcPefg89jKUX9CL9/38=
## =q/Fa
## -----END PGP MESSAGE-----
```

Test of arbitrary binary format with loader function

Note that `echo=FALSE` is quite important here. Perhaps I should force this on all data chunks?

```
plot(1:2,1:2,type="n")
rasterImage(img,1,1,2,2)
```


2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
1:2