Markdown / knitr in Rstudio

February 2016

Preliminaries

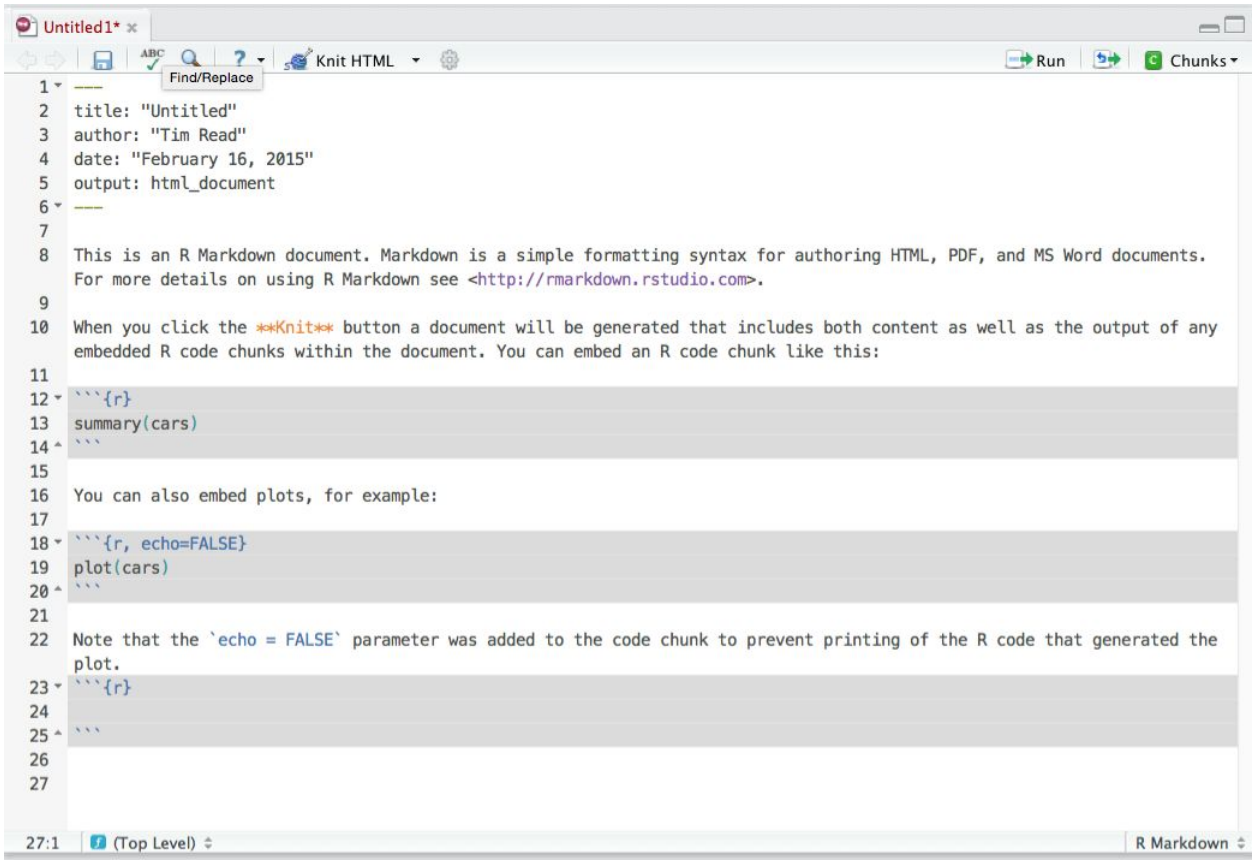
This was written using RStudio version 0.99.489 on Mac OS X and R vers 3.2.3
Load the R *knitr* and *markdown* packages (install using the package manager if you need).

Why Markdown and knitr?

Allows to you present your analysis as a report, with code and output that can be viewed as html, PDF or Word doc or even published directly to the web (http://rpubs.com). This is a valuable tool for Reproducible Research.

Rendering

Open a new R markdown file using the <File> dialog. You should see a document that looks something like this.



Now render the document into html by using the 'knitHTML' command and choosing the html option. What just happened?

Now, change the main title and add (using your new knowledge of markdown) some headings above each of the code chunks, and rerender.

Also, you can play around with the settings (wheel icon next to 'knitHTML') to add table of contents and change the look and feel of the report.

Code Chunks

There are quite a lot of options to control the behaviour of code chunks. See this link http://rmarkdown.rstudio.com/authoring_rcodechunks.html

One thing you will want to do a lot is format the output of dataframes. The kable method is one way to do this. Try adding,

````
```{r, results='asis'}
knitr::kable(mtcars)
```
````

Note - it is even possible to render python scripts using knitr! Try,

````
```{r engine = 'python'}
x = 'hello, python world!'
print(x)
```
````

Not sure I would recommend for very long code but it is still a nice feature to have.

Session Info

It is customary to put the following command in the last chunk

```
sessionInfo(package = NULL)
```

What does this do?
Why would this be useful?

Caching

When you are using this for your own projects you will eventually run into a problem when the code inside a chunk is taking a long time to execute. Since the process of creating the report is iterative, you may find yourself wasting a long time waiting for the same slow code to execute over and over again. This can be solved by caching, which means that the results are of the chunk are stored after the first time it is run, and then loaded into to the renderer the next time through. This is activated by adding inside the curly brackets

```
cache = TRUE
```

Another alternative is to use

```
eval = FALSE
```

Which means the chunk won't evaluate (run).
Caution is required as caching and skipping evaluation could have ramifications for the rest of the code. When in doubt, read the manual for details.

```
http://yihui.name/knitr/options
```

Exercise

Take the plain text file 'pubmed_coauthors.txt', copy and paste into a Rmarkdown document in Rstudio and format it so that it resembles this report 'pubmed_coauthors.html'. You will have to install the useful *dplyr* and *RISmed* packages if they are not already installed. (Obviously, the

session info won't be exactly the same as the html document, which was generated on my computer). The text is pasted below:

```
title: "Getting recent co-authors from Pubmed"
author: "Tim Read"
date:

Exersise 1 Generating a non-redundant list of coathors

This is a problem I had to solve for an NSF application.  The solution was
inspired by this blogpost
http://davetang.org/muse/2013/10/31/querying-pubmed-using-r/.

The key software is using the RISmed library, which allows you to access the
NIH EUtils API Application Package Interface
https://en.wikipedia.org/wiki/Application_programming_interface.  See this
blog.http://www.ncbi.nlm.nih.gov/books/NBK25500/

First, open the libraries and load the query

library("RISmed")
library("dplyr")
res <- EUtilsSummary('read td', type='esearch', db='pubmed', mindate='2013',
maxdate='2016')
fetch <- EUtilsGet(res)


Now use a for-loop to iteratively merge the first two columns####


a_list = NULL
for (i in 1:length(Author(fetch))){
  if (i == 1) {a_list <- Author(fetch)[[1]][,1:2]}
  else {
    a_list <- dplyr::union(a_list,Author(fetch)[[i]][,1:2])
  }
}

print(a_list)

Not bad: a lot of co-authors.

Exercise 2 - find and plot references containing the bacterium 'pandoraea' over
a multi-year period
```

```
Adapted from this blog entry
http://www.r-bloggers.com/how-to-search-pubmed-with-rismed-package-in-r/

#first how many total articles containing keyword?
res3 <- EUtilsSummary('pandoraea', type='esearch', db='pubmed')
summary(res3)
fetch3 <-  EUtilsGet(res3)
y <- YearPubmed(fetch3)

library(ggplot2)
date()
count<-table(y)
count<-as.data.frame(count)
names(count)<-c("Year", "Counts")
# num <- data.frame(Year=count$Year, Counts=cumsum(count$Counts))
# num$g <- "g"
# names(num) <- c("Year", "Counts", "g")
q <- ggplot(data=count, aes(x=Year, y=Counts))
q <- q +  geom_bar(stat="identity")
q <- q + ggtitle(paste("PubMed articles containing the word pandoraea")) +
     ylab("Number of articles") +
     xlab(paste("Year in Query date: ", Sys.time(), sep="")) +
     labs(colour="") +
     theme_bw()
q

sessionInfo()
```

For extra fun, try changing some of the search terms.

Further exploring

RISmed - a fully featured package for downloading from NCBI databases: take a further look -
http://cran.r-project.org/web/packages/RISmed/index.html,

Some examples of using RISmed
https://freshbiostats.wordpress.com/2013/12/03/analysis-of-pubmed-search-results-using-r/

Create a co-author network (with RISmed and other packages):
http://www.matthewmaenner.com/blog/2013/09/11/creating-co-author-networks-with-pubmed-r-and-gephi/

Mistune - a Markdown renderer for python code : http://mistune.readthedocs.org