An example R Markdown file

Illustrating use of R, bash, and Python code chunks

Christopher Paciorek

February 2022

1) How to generate a document from this file

From within R, you can run the document through the either the *rmarkdown* or *knitr* package for R to generate an html file, or through the *rmarkdown* package to generate PDF or Word (the latter being useful at times but hopefully avoidable).

Alternatively, start R and run the desired line from amongst following in R:

```
library(rmarkdown); render('demo.Rmd', 'pdf_document')
library(rmarkdown); render('demo.Rmd', 'html_document')
library(rmarkdown); render('demo.Rmd', 'word_document')
library(knitr); knit2html('demo.Rmd')
```

Or in RStudio, click on the 'Knit' pull-down menu and choose to knit to HTML, PDF, or Word (for R Markdown).

Alternatively, from the UNIX command line, run one of these:

```
Rscript -e "library(rmarkdown); render('demo.Rmd', 'pdf_document')"  # PDF
Rscript -e "library(rmarkdown); render('demo.Rmd', 'html_document')"  # HTML
Rscript -e "library(rmarkdown); render('demo.Rmd', 'word_document')"  # Word
Rscript -e "library(knitr); knit2html('demo.Rmd')"  # HTML alternative
```

2) Some basic Markdown formatting

Here's an *introduction* to our **critical** discovery. Here we have some code to display inline but not evaluate: `exp(7)` and we can embed the code in a static code block as follows:

```
a = 7 %% 5
b = exp(a)
```

This document will focus on embedding math and code and not on standard Markdown formatting. There are lots of sources of information on Markdown. RStudio has good information on R Markdown (including Markdown formatting).

For documents whose output format is HTML, you can use HTML formatting within your Markdown-based text.

3) Embedding equations using LaTeX

This can be done with the following syntax. Note that you can't have a space after the initial $ for the inline equations.

Here is an inline equation $f(x) = \int f(y, x)dy$.

Here's a displayed equation

$$f_\theta(x) = \int f_\theta(y, x)dy.$$

4) Embedding R code

Here's an R code chunk

```
a <- c(7, 3)
mean(a)
```

```
## [1] 5
```

```
b <- a + 3
mean(b)
```

```
## [1] 8
```

Here's another chunk:

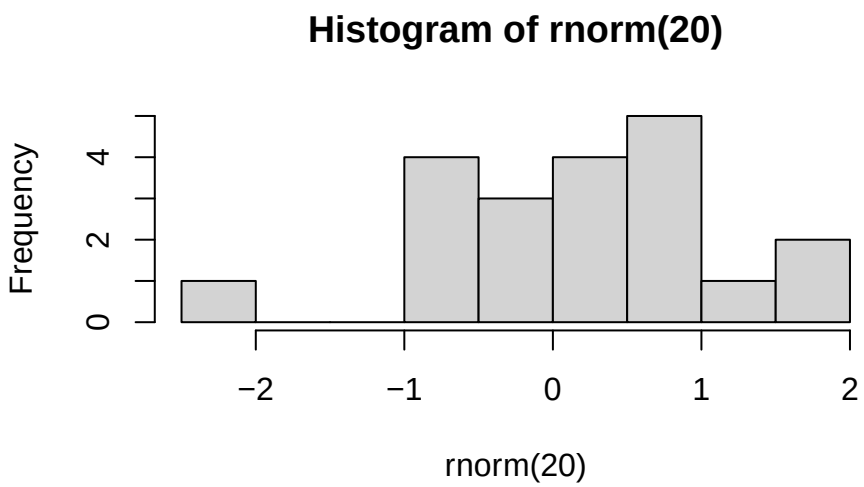
```
mean(b)
```

```
## [1] 8
```

When running R code, output is printed interspersed with the code, as one would generally want. Also, later chunks have access to result from earlier chunks (i.e., state is preserved between chunks).

Let's make a plot:

```
hist(rnorm(20))
```



And here's some inline R code: What is 3 plus 5? 8.

You have control over whether code in chunks is echoed into the document and evaluated using the `include`, `echo`, and `eval` tags.

```
## This code is not printed in the document, but results of evaluating the code are printed.
```

```
cat("This code is not evaluated, but the code itself is printed in the document.")
```

Results of intensive calculations can be saved using the `cache=TRUE` tag so they don't need to be rerun every time you compile the document.

```r
a <- mean(rnorm(5e7))
a
```

```
## [1] -1.666068e-05
```

You can use R variables to control the chunk options. Note that the variable `myControlVar` is defined in the first chunk of this document.

```r
print("hi")
```

An alternative, nice way to specify chunk options is within the chunk, like this:

```r
cat("This code is printed in the document, but the code is not evaluated.")
```

5) Embedding bash and Python code

5.1) bash

A bash chunk:

```bash
ls -l
df -h
cd /tmp
pwd
```

```
## total 2994
## drwxr-sr-x 6 paciorek scfstaff      6 Nov 17 13:21 assets
## drwxr-sr-x 2 paciorek scfstaff     10 Feb  1 16:49 cache
## -rw-r--r-- 1 paciorek scfstaff    391 Jan 31 15:16 _config.yml
## -rw-r--r-- 1 paciorek scfstaff 282459 Feb  2 15:41 demo-bash.html
## -rw-r--r-- 1 paciorek scfstaff   8083 Feb  2 15:41 demo-bash.ipynb
## -rw-r--r-- 1 paciorek scfstaff  48673 Feb  2 15:42 demo-bash.pdf
## drwxr-sr-x 6 paciorek scfstaff      6 Jan 27 16:27 demo_cache
## -rw-r--r-- 1 paciorek scfstaff  17894 Jan 27 16:53 demo.docx
## drwxr-sr-x 6 paciorek scfstaff      6 Jan 27 16:27 demo_files
## -rw-r--r-- 1 paciorek scfstaff 660776 Feb  2 15:19 demo.html
## -rw-r--r-- 1 paciorek scfstaff  24939 Feb  1 16:55 demo.lyx
## -rw-r--r-- 1 paciorek scfstaff 259403 Feb  2 15:18 demo.pdf
## -rw-r--r-- 1 paciorek scfstaff 284419 Feb  2 15:44 demo-python.html
## -rw-r--r-- 1 paciorek scfstaff   6936 Feb  2 15:44 demo-python.ipynb
## -rw-r--r-- 1 paciorek scfstaff  49626 Feb  2 15:44 demo-python.pdf
## -rw-r--r-- 1 paciorek scfstaff   3662 Jan 27 15:53 demo-python.qmd
## -rw-r--r-- 1 paciorek scfstaff    252 Jan 31 17:05 demo.R
## -rw-r--r-- 1 paciorek scfstaff 321394 Feb  2 15:54 demo-R.html
## -rw-r--r-- 1 paciorek scfstaff 124604 Feb  2 15:54 demo-R.ipynb
## -rw-r--r-- 1 paciorek scfstaff  13004 Feb  2 15:19 demo.Rmd
## -rw-r--r-- 1 paciorek scfstaff  10853 Nov  6  2019 demo.Rmd.save
## -rw-r--r-- 1 paciorek scfstaff  12462 Feb  1 16:48 demo.Rnw
## -rw-r--r-- 1 paciorek scfstaff  59736 Feb  2 15:54 demo-R.pdf
## -rw-r--r-- 1 paciorek scfstaff  12890 Feb  1 16:14 demo.Rtex
## drwxr-sr-x 2 paciorek scfstaff      5 Jul 30  2015 figure
## drwxr-sr-x 2 paciorek scfstaff      3 Nov 19 13:39 _includes
## -rw-r--r-- 1 paciorek scfstaff   7874 Feb  2 14:46 index.md
## drwxr-sr-x 2 paciorek scfstaff      4 Feb  3 11:48 _layouts
## -rw-r--r-- 1 paciorek scfstaff    475 Feb  3 11:57 Makefile
## -rw-r--r-- 1 paciorek scfstaff   2082 Feb  3 11:57 python_in_RStudio.md
```

```
## -rw-r--r-- 1 paciorek scfstaff 171796 Feb  2 16:11 python_in_RStudio.pdf
## -rw-r--r-- 1 paciorek scfstaff   2013 Feb  2 16:11 python_in_RStudio.Rmd
## -rw-r--r-- 1 paciorek scfstaff    773 Feb  3 11:53 README.md
## -rw-r--r-- 1 paciorek scfstaff  14902 Jul 30  2015 refs.bib
## drwxr-sr-x 2 paciorek scfstaff      7 Nov 22 15:21 _sass
## -rw-r--r-- 1 paciorek scfstaff   7167 Jul 17  2015 test-line-formatting.Rnw
## Filesystem                            Size  Used Avail Use% Mounted on
## /dev/sda1                             120G   21G   94G  18% /
## udev                                  7.8G     0  7.8G   0% /dev
## tmpfs                                 7.8G  411M  7.4G   6% /dev/shm
## tmpfs                                 1.6G   16M  1.6G   1% /run
## tmpfs                                 5.0M  4.0K  5.0M   1% /run/lock
## tmpfs                                 7.8G     0  7.8G   0% /sys/fs/cgroup
## /dev/sda2                             240G   37G  191G  16% /var
## /dev/sda3                             240G  142G   86G  63% /var/tmp
## /dev/loop1                            142M  142M     0 100% /snap/chromium/1708
## /dev/loop0                            142M  142M     0 100% /snap/chromium/1691
## /dev/loop2                            165M  165M     0 100% /snap/gnome-3-28-1804/161
## /dev/loop4                             56M   56M     0 100% /snap/core18/2128
## /dev/loop7                             33M   33M     0 100% /snap/snapd/12704
## /dev/loop5                             66M   66M     0 100% /snap/gtk-common-themes/1515
## /dev/loop3                            163M  163M     0 100% /snap/gnome-3-28-1804/145
## /dev/loop6                             56M   56M     0 100% /snap/core18/2074
## /dev/loop8                             33M   33M     0 100% /snap/snapd/12883
## /dev/sda5                             286G  6.1G  265G   3% /tmp
## sauron.berkeley.edu:/pool0/accounts    12T  6.8T  4.9T  59% /accounts
## tmpfs                                 1.6G   84K  1.6G   1% /run/user/3189
## sauron.berkeley.edu:/pool0/system     5.0T  3.8T  1.3T  76% /system
## oz.berkeley.edu:/pool0/scratch         84T   26T   58T  31% /scratch
## /tmp
```

Unfortunately, output from bash chunks occurs after all the code is printed. Also, state is not preserved between chunks.

We can see that state is not preserved here, where the current working directory is NOT the directory that we changed to in the chunk above.

```
pwd  # result would be /tmp if state were preserved
```

```
## /accounts/gen/vis/paciorek/staff/tutorials/tutorial-dynamic-docs
```

Inline bash code won't work: `bash wc demo.Rmd`, unlike with R code.

5.2) Embedding Python code

You can embed Python code. As with R, state is preserved so later chunks can use objects from earlier chunks.

```
import numpy as np
x = np.array((3, 5, 7))
print(x.sum())
```

```
## 15
```

```
x.min()    # this will print with more recent versions of rmarkdown
```

```
## 3
```

```
try:
        print(x[0])
except NameError:
       print('state is not preserved: x does not exist')
```

```
## 3
```

There is no facility for inline Python code: `python print(3+5)`

6) Reading code from an external file

It's sometimes nice to draw code in from a separate file. Before invoking a chunk, we need to read the chunks from the source file, which contains the chunks tagged with some special formatting. Note that a good place for reading the source file via `read_chunk()` is in an initial setup chunk at the beginning of the document.

```
a <- 7
cat("a is ", a, ".\n", sep = "")
```

```
## a is 7.
```

```
a <- 9
cat("Now, a is ", a, ".\n", sep = "")
```

```
## Now, a is 9.
```

7) Formatting of long lines of code and of output

7.1) R code

Having long lines be nicely formatted and other aspects of formatting can be a challenge. Also, results can differ depending on your output format (e.g., PDF vs. HTML). In general the code in this section will often overflow the page width in PDF but not in HTML, but even in the HTML the line breaks may be awkwardly positioned.

Here are some examples that overflow in PDF output.

```
b <- "Statistics at UC Berkeley: We are a community engaged in research and education in probability an
## Statistics at UC Berkeley: We are a community engaged in research and education in probability and s

## This should work to give decent formatting in HTML but doesn't in PDF.
cat(b, fill = TRUE)
```

```
## Statistics at UC Berkeley: We are a community engaged in research and education in probability and s
```

```
vecWithALongName = rnorm(100)
a = length(mean(5 * vecWithALongName + vecWithALongName - exp(vecWithALongName) + vecWithALongName * ve
a = length(mean(5 * vecWithALongName + vecWithALongName)) # this is a comment that goes over the line b
a = length(mean(5 * vecWithALongName + vecWithALongName - exp(vecWithALongName) + vecWithALongName, na.
```

In contrast, long output is usually fine, even in PDF.

```
rnorm(30)
```

```
##  [1] -0.969403453  1.532418315  0.947017779 -0.018912299  0.576774020
##  [6] -1.075577378  0.748625017 -0.479714201  0.598189960  0.767062074
## [11] -1.451185453  0.588824254  0.442839312  0.637782742  0.009485898
## [16]  0.168970871 -0.237679050 -1.171665598  0.752141969 -0.135204250
## [21] -1.056915733  0.501191065  0.158616517 -1.651353734  0.540141303
```

```
## [26] -0.915850243  0.180769022 -1.125076489 -0.502575556  1.263204251
```

Adding the `tidy=TRUE` chunk option and setting the width (as shown in the Rmd version of this document) can help with long comment lines or lines of code, but doesn't help for some of the cases above.

```
## Long strings and long comments:

b <- "Statistics at UC Berkeley: We are a community engaged in research and education in probability an
## Statistics at UC Berkeley: We are a community engaged in research and
## education in probability and statistics. In addition to developing
## fundamental theory and methodology, we are actively

## This should work to give decent formatting in HTML but doesn't in PDF:

cat(b, fill = TRUE)
```

```
## Statistics at UC Berkeley: We are a community engaged in research and education in probability and s
```

```
## Now consider long lines of code:

vecWithALongName <- rnorm(100)
a <- length(mean(5 * vecWithALongName + vecWithALongName - exp(vecWithALongName) +
    vecWithALongName * vecWithALongName, na.rm = TRUE))
a <- length(mean(5 * vecWithALongName + vecWithALongName))  # this is a comment that goes over the line
a <- length(mean(5 * vecWithALongName + vecWithALongName - exp(vecWithALongName) +
    vecWithALongName, na.rm = TRUE))  # this is a comment that goes over the line by a good long long l
```

To address the problems seen above, sometimes you can format things manually for better results. You may need to tag the chunk with `tidy=FALSE`, but I have not done that here.

```
## Breaking up a string:

b <- "Statistics at UC Berkeley: We are a community engaged in research
 and education in probability and statistics. In addition to developing
fundamental theory and methodology, we are actively"

## Breaking up a comment:

## Statistics at UC Berkeley: We are a community engaged in research and
## education in probability and statistics. In addition to developing
## fundamental theory and methodology, we are actively

## Breaking up code lines:

vecWithALongName = rnorm(100)
a <- length(mean(5 * vecWithALongName + vecWithALongName - exp(vecWithALongName) +
    vecWithALongName * vecWithALongName, na.rm = TRUE))
a <- length(mean(5 * vecWithALongName + vecWithALongName)) # this is a comment that
    ## goes over the line by a good long ways
a <- length(mean(5 * vecWithALongName + vecWithALongName - exp(vecWithALongName) +
    vecWithALongName, na.rm = TRUE)) # this is a comment that goes over the line
    ## by a good long long long long long long long ways
```

7.2) bash code

In bash, we have similar problems with lines overflowing in PDF output, but bash allows us to use a backslash to break lines of code. However that strategy doesn't help with long lines of output.

```
echo "Statistics at UC Berkeley: We are a community engaged in research and education in probability and

echo "Second try: Statistics at UC Berkeley: We are a community engaged \
in research and education in probability and statistics. In addition to \
developing fundamental theory and methodology, we are actively" \
>> tmp.txt

cat tmp.txt
```

```
## Statistics at UC Berkeley: We are a community engaged in research and education in probability and s
## Second try: Statistics at UC Berkeley: We are a community engaged in research and education in probab
```

We also have problems with long comments, so we would need to manually format them.

Here is a long comment line that overflows in PDF:

```
# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfs
```

Instead manually break the comment into multiple lines:

```
# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla
# lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfsa lajdfa
# adlfjaf jkladf afdl
```

7.3) Python code

In Python, there is similar trouble with lines overflowing in PDF output too.

```
# This overflows the page:

b = "asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkaj
print(b)
```

```
## asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdf
```

```
# This code overflows the page:

zoo = {"lion": "Simba", "panda": None, "whale": "Moby", "numAnimals": 3, "bear": "Yogi", "killer whale"
print(zoo)
```

```
## {'lion': 'Simba', 'panda': None, 'whale': 'Moby', 'numAnimals': 3, 'bear': 'Yogi', 'killer whale': '
```

To fix the issue, we can manually break the code into multiple lines, but long output still overflows.

```
zoo = {"lion": "Simba", "panda": None, "whale": "Moby",
       "numAnimals": 3, "bear": "Yogi", "killer whale": "shamu",
       "bunny:": "bugs"}
print(zoo)
```

```
## {'lion': 'Simba', 'panda': None, 'whale': 'Moby', 'numAnimals': 3, 'bear': 'Yogi', 'killer whale': '
```

Long comments overflow as well, but you can always manually break into multiple lines.

```
# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfs

# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad
# kljafda kaljdf afdlkja lkajdfsa lajdfa adlfjaf jkladf afdl
```

8) References

We'll just see how you use BibTeX style references. Banerjee et al. (2008) proposed a useful method. This was confirmed (Cressie and Johannesson 2008).

Note the indication of the `refs.bib` file in the initial lines of this document so that the bibliographic information for these citations can be found.

The list of references is placed at the end of the document. You'd presumably want a section header like this:

Literature cited

Banerjee, S., A.E. Gelfand, A.O. Finley, and H. Sang. 2008. "Gaussian Predictive Process Models for Large Spatial Data Sets." *Journal of the Royal Statistical Society B* 70 (4): 825–48.

Cressie, N., and G. Johannesson. 2008. "Fixed Rank Kriging for Very Large Spatial Data Sets." *Journal of the Royal Statistical Society B* 70 (1): 209–26.