An example Rtex file
Illustrating use of R, bash, and Python code chunks

Christopher Paciorek

August 2015

1 How to generate a document from this file

In bash, you can run this document through the *knitr* package for R to generate a PDF.

```
Rscript -e "library(knitr); knit2pdf('demo.Rtex')"
```

Or, start R and run this:

```
library(knitr); knit2pdf('demo.Rtex')
```

I don't know of a way to generate HTML directly, but there are tools for converting between various file formats, such as *pandoc*.

2 LaTeX

This document will focus on embedding code and not on standard LaTeX. For a quick introduction to LaTeX, see our LaTeX tutorial.

The `knit2pdf` command processes the Rtex format, evaluating the R code chunks, and creating a standard LaTeX file with the code snippets and output created by the code properly formatted as standard LaTeX.

3 Embedding equations using LaTeX

The knitr Rtex format is built on LaTeX, so you can just use LaTeX functionality.

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
```

Unfortunately, we need to use the *noindent* command to prevent the text after a chunk from being considered as a new paragraph. Here's another chunk:

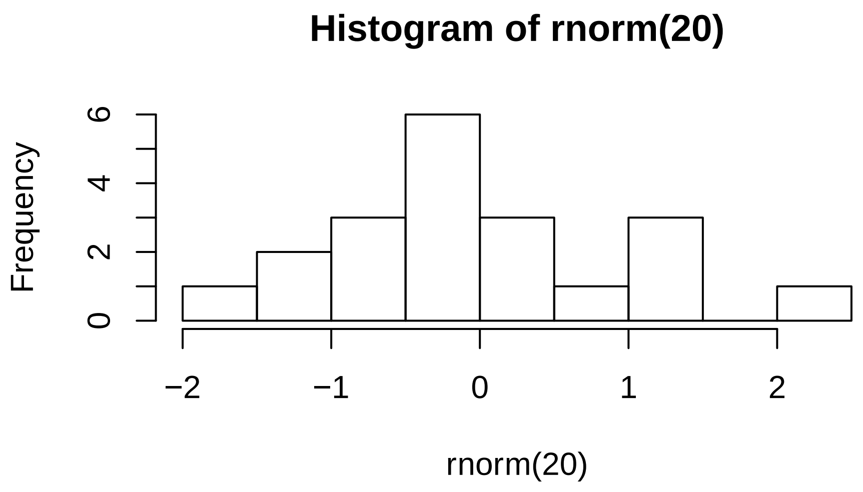
```
mean(b)

## [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")
```

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

```
mean(rnorm(5e7))

## [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.

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

5 Embedding bash and Python code

5.1 bash

A bash chunk:

```
df -h
cd /tmp
pwd

## Filesystem                              Size  Used Avail Use% Mounted on
## /dev/sda1                               120G   18G   96G  16% /
## udev                                    7.8G     0  7.8G   0% /dev
## tmpfs                                   7.9G  454M  7.4G   6% /dev/shm
## tmpfs                                   1.6G   26M  1.6G   2% /run
## tmpfs                                   5.0M  4.0K  5.0M   1% /run/lock
## tmpfs                                   7.9G     0  7.9G   0% /sys/fs/cgroup
## /dev/sda5                               286G  2.0G  269G   1% /tmp
## /dev/sda2                               240G   85G  143G  38% /var
## /dev/sda3                               240G  158G   70G  70% /var/tmp
## sauron.berkeley.edu:/pool0/system       4.5T  2.9T  1.7T  65% /system
## tmpfs                                   1.6G  324K  1.6G   1% /run/user/3189
## sauron.berkeley.edu:/pool0/accounts      16T  5.6T  9.7T  37% /accounts
## sauron.berkeley.edu:/pool0/scratch       23T   20T  3.5T  85% /scratch
## /tmp
```

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

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

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

Inline bash code won't work.

5.2 Embedding Python code

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

## 15

x.min()  # note not printed

## 3
```

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

## 3
```

As with bash chunks, all output occurs after the code is printed. Also, state is not preserved between chunks. Finally, you need explicit print statements; you won't see what would normally be printed to the screen.

There is no facility for inline Python code evaluation.

6 Reading code from an external file

It's sometimes nice to draw code in from a separate file. 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.

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

# this should work but often doesn't
cat(b, fill = TRUE)

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

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

# long output usually is fine
rnorm(100)

##   [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
##  [31]  0.305884237  1.179978412  0.335135837  0.367916350  1.083923402
##  [36] -1.184164989  1.537729698  0.361302447 -0.172584477 -0.213126301
##  [41] -0.098800217 -3.284717302  0.136839013  0.636921042  1.353787943
##  [46]  0.181222170  1.369503117  0.011455738  0.730555444  0.373799109
##  [51] -2.202612634 -1.749906270 -0.113214405  0.740377591 -2.639445915
##  [56]  1.239964362 -1.008510432  0.037322296  0.648640173  0.474265300
##  [61]  0.253960175 -0.645791926  1.634776833 -0.644956138 -1.150062767
##  [66] -0.550337018 -0.508750139 -0.782182176 -0.001985901 -0.976521061
##  [71]  0.615158496 -1.020881413 -1.153291895 -0.352463722 -0.963285313
##  [76] -0.408517582  0.276794454 -0.199733183  1.281162691  0.369146818
##  [81] -0.963669212 -0.066022644  0.523984116  1.949630076  0.093499705
##  [86]  0.571681512 -0.355790910 -1.205270217  0.058748521  0.538841352
##  [91] -1.111665772  1.962392444  2.089893978 -0.956068574 -1.697766206
##  [96] -0.093768237  1.241127762  0.588783071  0.235587281 -1.150094577
```

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

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.

```
# the following long comment line is not broken in my test:
# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfsa

# instead manually break it:
# 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 in some contexts
b = "asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkaj
print(b)

# this code overflows the page in some contexts

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

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

# instead manually break the code

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

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

# long comments overflow too
# asdl lkjsdf jklsdf kladfj jksfd alkfd klasdf klad kla lakjsdf aljdkfad kljafda kaljdf afdlkja lkajdfs

# and the long output that will appear next in the resulting document (produced from the evaluation of 

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

8 References

Here's how to use BibTeX style references. Banerjee et al. [2008] proposed a useful method. This was confirmed [Cressie and Johannesson, 2008].

Note the use of the bibliography keyword below to indicate the *refs.bib* file as the source of the bibliographic information for the references above.

The list of references is placed at the end of the document.

References

S. Banerjee, A.E. Gelfand, A.O. Finley, and H. Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society B*, 70(4):825–848, 2008.

N. Cressie and G. Johannesson. Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society B*, 70(1):209–226, 2008.