Homework 12. Due by 5pm on Wednesday 12/2.

Integrating text and code: dynamic documents via knitr

As discussed in class, there are numerous advantages to writing a statistics paper in such a way that the tables, figures and other quantitative results are automatically generated from chunks of code included in the document. These include:

1. Changing the data. Questions like "How stable are my conclusions? What happens to all my figures and tables if I omit the 5 smallest states from my panel of 50 states?" can be rapidly answered. The easier it is to investigate new analyses, the more things you try.
2. Effective collaboration. All coauthors can read, run and modify all the code that produced the figures in the current version of a circulated draft.
3. Debugging. If your adviser asks "How exactly did this number get produced?" you can give a rapid, precise and accurate answer.
4. Updating. If you are presenting code (e.g., lecture notes) and you want to make changes where necessary for a new software version, you simply re-run the document.
5. Revisions. 4 months after you submitted the paper, when the referee reports come back, you will be glad if you have your work organized in this way!

This homework asks you to carry out a few manipulations on a knitr document. Specifically, we consider the task of reproducing a recent paper prepared as a knitr document. This paper is not to be considered as a model of high-quality reproducibility, but rather as an example of some modest efforts toward the goal of reproducibility.

Please carry out the following tasks and report back to me by email if you successfully complete them or hit obstacles.

1. Download the files in `http://dept.stat.lsa.umich.edu/~ionides/810/knitr_example`
2. Software gets out of date fast! The code is broken if you use the current version of the R package `pomp`. The computations used `pomp_0.53-1` which you can download and install, as in homework 10, from

   `https://cran.r-project.org/src/contrib/Archive/pomp/`

   (The knitr file and/or the paper should have said the versions of R and `pomp` used, but did not.)
3. Try various ways to generate the pdf file from the Rnw file. Specifically,

(a) In an R session, with the knitr package installed,

```
require(knitr)
knit("if2.Rnw")
```

Then, run Latex on the resulting if2.tex file

(b) From a Linux terminal prompt,

```
Rscript -e "require(knitr); knit('if2.Rnw')"
```

Then, run Latex on the resulting if2.tex file

(c) Using the make program (`http://kbroman.org/minimal_make`). On a Linux machine, making sure that the HW12 files (including Makefile) are all in the current working directory,

```
make if2.pdf
```

(d) Using RStudio. Open if2.Rnw in RStudio and click 'Compile pdf.' This failed to run for me. I suspect that my file contains some deprecated knitr syntax, included for back-compatibility with Sweave and no longer supported by RStudio. I'm not much of an RStudio user, but RStudio seems to be increasingly prevalent. Optionally, if you can debug this, please let me know!

4. Question: which of these approaches could you get to work? What are the advantages and disadvantages of each approach? Which could be used on flux?

5. Look through if2.Rnw, focusing on the code chunks. You don't really have to understand much about knitr to do this, but if you like you can browse `http://yihui.name/knitr/`.

   (a) Notice the flag `TEST=TRUE`. This runs a quick version of the code, to check for syntax errors before running a big job.

   (b) If you set `TEST=FALSE`, the code will take several hours on 100 flux cores. If you were to do that (which you are not expected to do) you should be able to perfectly match the figures in the published paper, at `http://www.pnas.org/content/112/3/719`. The R code published as supporting information on the journal website, was extracted from the Rnw file by

   ```
   knitr::purl("if2.Rnw")
   ```

   (c) How is caching done? Specifically, how does the code decide whether to re-run the long computations? Hint: this is done explicitly, without relying on knitr's built-in caching system.

   (d) Notice how the random number generator seed is set to give reproducible results (e.g., search for "seed" in if2.Rnw). Subtle problems can arise when setting seeds for parallel computations. Can you think of any? Perhaps the solution here should be improved.