

Homework 13. Due by 5pm on Thursday 12/3.

A workflow for reproducible statistical research: combining Latex and R with knitr

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., a lab presentation) 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!

Rmarkdown and Jupyter are convenient platforms for exploratory investigations, but knitr is well placed to generate Latex for publication-quality pdf articles. This homework investigates a workflow (meaning a set of tools and procedures that together get research done effectively) used for the research project at

https://github.com/ionides/bagged_filters

The code is somewhat complex and involves various features that may be new to you. You are welcome to ask your peers for help if you get stuck. This is an open-ended assignment pulling together things we have studied earlier, and some new material. You are welcome to decide for yourself how long to spend on it. Edit the file `810f20/hw13.Rnw` with your answers, build a pdf and submit to Canvas.

1. Clone the `bagged_filters` git repository to your laptop, as in Homework 9. There are three pdf files:
 - (a) `ms.pdf`, the main article.
 - (b) `si.pdf`, an online supplement to be published with the article.

(c) `ext/ext.pdf`, unpublished results and ideas saved during the research.

We will focus on `ms.pdf`. You do not have to report anything for this step.

2. The source file for `ms.pdf` is `ms.Rnw`, a file in R noweb format designed to be run by `knitr()`. Rstudio runs `knitr()` automatically when you ask it to build from an Rnw file, but for workflows based on text commands it is good practice to call `knitr()` directly from R. Rnw files simply combine chunks of \LaTeX with chunks of R code.

Have you used Rnw format before, either through Rstudio or not?

YOUR ANSWER HERE

3. There are various ways to compile `ms.Rnw` to `ms.pdf`. All of them need the necessary R packages, which you may need to install on your laptop or greatlakes or anywhere else you try running the code. Most of the R packages required are on CRAN, but the `spatPomp` package is not. It is at

<https://github.com/kidusasfaw/spatPomp>

There are different ways to install an R package from source code. A direct approach is to clone the repository and then run

```
R CMD INSTALL spatPomp
```

in the directory where `spatPomp` is located. Alternatively, you can use `install_github()` from the `devtools` package. Note that before installing `spatPomp` you will need to have `pomp` installed, for which you may need to consult the instructions at

<https://kingaa.github.io/pomp/install.html>.

The installation of `pomp` is nontrivial because this package carries out compilation of C code, so you need to have a C compiler installed and talking properly to R. Time spent figuring this out is not entirely wasted.

Now, in an R session running in the `bagged_filters` directory, you can run

```
library(knitr)
knit("ms.Rnw")
```

If all is well, this will generate a file `ms.tex` which can be used to produce `ms.pdf` by running `pdflatex`. Likely, issues will arise that need to be solved to get this working. Spend a reasonable amount of time trying to get this working. It is okay if you cannot get the code to run, since many of the questions do not depend on this. Report on whether you were successful, what problems you overcame, and where you got stuck.

YOUR ANSWER HERE.

4. Have you used `make` before? ([https://en.wikipedia.org/wiki/Make_\(software\)](https://en.wikipedia.org/wiki/Make_(software))) This is a standard tool for organizing scientific coding projects, and it is installed by default on Mac and Linux systems. The `bagged_filters` directory has a Makefile, so you can run

```
make ms.pdf
```

at a terminal prompt to build `ms.pdf` from `ms.Rnw`. This just runs `knitr` followed by `pdflatex` so it cannot work unless the separate steps are working. Try this, and report briefly

YOUR ANSWER HERE.

5. The manuscript can also be built on `greatlakes`, and this is appropriate for a production version having numerical calculations too extensive for a laptop. Identify the critical lines of code to enable the program to run on either `greatlakes` or a laptop.

YOUR ANSWER HERE

Optionally, work on building `ms.tex` on `greatlakes`, by running

```
sbatch ms.sbat
```

This requires cloning the git repository to `greatlakes` and installing all necessary R packages locally in your `greatlakes` account.

6. Writing a reasonably large reproducible document combining text and code, you cannot avoid the issue of caching. You do not want to re-run all computations each time you edit any text in the document, so you must save (i.e., cache) results that do not need to be recomputed. Ideally, when we edit code we would re-run only the partial results that have changed as a consequence of the edit. Sadly, it is intractable to automate this in a foolproof way. The `knitr` code chunk option `cache=TRUE` re-runs a code chunk if that particular chunk is edited. It is necessary to delete all cached files (e.g., `rm -rf cache`) occasionally to rebuild the cache correctly. In `ms.Rnw`, the `stew()` function from the `pomp` package is used to give additional manual control of caching the most time-consuming results.

Have you had any prior experience working with `cache` on reproducible documents?

YOUR ANSWER HERE.

7. For debugging, it is helpful to have a quick version of the code which can be run to check for errors before starting a long, expensive computing job. Setting `run_level=1` in `ms.Rnw` gives a version of the code that runs in a few seconds. If you can successfully compile `ms.tex` or `ms.pdf`, try deleting all the cached results from `run_level=1` by

```
rm -rf *_1
```

Then if you knit the document you will run quick versions of all the computations. Did that work for you?

If you are new to Linux/Unix, work out what the parts of the command `rm -rf *_1` do. The `f` flag may not be necessary for you.

YOUR ANSWER HERE.

8. Workflows for writing manuscripts are build up over years, borrowed, shared, and modified for different purposes and evolving technologies. Compare this workflow with the range of techniques you already use.

YOUR ANSWER HERE.

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

YOUR ANSWER HERE.