An Introduction to Sweave
Computing for Research I

Elizabeth G Hill

10 April 2012

Adapted in part from a presentation by Frank Harrell, ENAR, 22 March 2010

Does this sound familiar?

- You need to modify simulations for a paper you've submitted to *Biometrics*. It has been six months since you last looked at these files. After locating the directory on your computer, you realize there are multiple data files, multiple analysis files, and multiple results files. You can't remember which ones are pertinent.
- You've been working for a week analyzing data for a collaborator. You've just completed the final report, and you receive an email requesting you rerun analyses based on a subset of the original data, and you need to have it done by tomorrow morning.
- You're re-running the primary analysis for your manuscript/dissertation/grant proposal, and the results don't match. You've spent hours trying to re-trace the steps taken to produce the original results, but to no avail.

The case of Dr. Anil Potti

- http://dukechronicle.com/article/cancer-research-questioned
- “Baggerly and Coombes ... found many of the details vague, which made it difficult to replicate Potti’s findings.”
- “One of the difficulties is that in many of the journal articles in this field, the descriptions of the methods that actually go into the papers... just don’t have enough detail to figure out exactly how the data were analyzed ...”
- “Baggerly and Coombes continued analyzing the research - spending about 1,500 hours in all - and found additional problems, including ones they said might jeopardize patient safety.”

Sources of non-reproducibility

- Copy and paste errors
- Analysis of ‘calculated’ variables
- Poor or absent documentation
- Multiple data manipulation steps and analysis tools with no audit trail
- Programming errors
- Pre-statistician data ‘normalization’
- Insufficient detail in scientific articles

Goals of reproducible research

- Be able to reproduce your own work
- Allow others to reproduce your work
- Reproduce report/paper/dissertation with a single command when changes occur in
  - software
  - operating system
  - graphics
  - data
  - analysis
- Save time
- Transparency

Sweave’s approach

- ‘Weave’ the analysis into the report (using Sweave)
- Upon execution, all pertinent analysis output is created and inserted into report
- Removes ‘cut and paste’ mentality
- A single file is maintained
- The report/paper/dissertation is fully reproducible
- The report/paper/dissertation is dynamic - regeneration not an issue

Report generation using Sweave

1. Generate an R ‘noweb’ (.Rnw) Sweave source file in a text editor (like WinEdt or Emacs). This will be written interactively with R.
2. Run the R ‘noweb’ (.Rnw) Sweave source file through the Sweave function in R. The command is `Sweave("<filename>.Rnw")`. This generates a LaTeX(.tex) file.
3. Run the LaTeX(.tex) file through a LaTeXcompiler to generate a DVI (.dvi), postscript (.ps), or PDF (.pdf) file.

Some noweb basics

Literate programming was introduced by Donald Knuth in the 1970s as an alternative programming approach that follows the flow and logic of the programmer's ideas rather than the language syntax of the computer. Noweb is a literate programming tool that combines program source code and corresponding documentation in a single file. Code and documentation are separated into *chunks*.

- @ denotes the start of documentation chunks. We will write our documentation in LaTeX.
- `<< options >>=` denotes the start of code chunks. We will write our code in `R`.

Tangling and weaving

There are two frontend functions for the Sweave system - `Sweave()` and `Stangle()`. Both are contained in the R package `utils` and are automatically downloaded when you install R.

- `Stangle()` is used to extract only the code chunks from the .Rnw file.
- `Sweave()` runs the code chunks and replaces them with the appropriate input/output.
- We will only need to run `Sweave()` to generate our reports.

Basic code chunk options

`label` This is a text label for the code chunk

- Use `label=` anywhere in code option.
- If listed first then no `label=` is required.
- If no label is provided, then the label for the code chunk is it's number.

`echo` logical (TRUE or FALSE)

- If not specified then default is `echo=TRUE`
- If set to `TRUE`, then R code is echoed into the LaTeXdocument.
- If set to `FALSE`, then R code is not echoed into the LaTeXdocument.

Basic code chunk options (cont.)

`fig` logical (TRUE or FALSE)

- If not specified then default is `fig=FALSE`
- If set to `TRUE`, then plot is included.
- If set to `FALSE`, then plot is not included.

`include` logical (TRUE or FALSE)

- If not specified then default is `include=TRUE`
- Set to `TRUE` for text output and `includegraphics` statements to be automatically generated.
- Set to `FALSE` when output should appear in a different place.

Basic code chunk options (cont.)

`results` character string

- If not specified then default is `results=verbatim`
- `results=verbatim` indicates the R output is included in a verbatim-like environment.
- `results=hide` means all output is completely suppressed, but code is executed during the weave.

Let's look at our example.

Some comments

- `\SweaveOpts{prefix.string=CFRIeg1}` means that all graphics files will have the prefix `CFRIeg1`.
- If you don't specify a prefix string for graphics files, then they are named `filename-chunkname.ps` (or .pdf).
- `\SweaveOpts{eps=TRUE,pdf=FALSE}` means the graphics files will be encapsulated postscript and not pdf.
- If you omit this option, then both .eps and .pdf files are created.
- `\Sexpr{}` means *S expression* and allows for evaluation of a scalar within a documentation chunk. Anything more complicated should be evaluated in a code chunk.
- `\verbatimtabinput{Sweave_example.Rnw}` produces the source code in the final report.