gompertzTest.R

```
require(doParallel)
# reads the list of nodes which have been
# allocated by the cluster queue manager
nodefile <- Sys.getenv("PBS_NODEFILE")
hostlist <- read.table(nodefile,skip=1,
        header=FALSE)

# builds a socket cluster using these nodes
cl <- makeCluster(c(as.character(hostlist$V1)),
        type='SOCK')
registerDoParallel(cl)

# a simple parallel for loop using using foreach
r <- foreach(1:100,.packages='pomp') %dopar% {
        pompExample(gompertz)
        sim <- simulate(gompertz)
}

stopCluster(cl)
save(r,file="sims.Rda")
```

runTest.pbs, part I

```
#!/bin/csh
##This names the job for the queueing system
#PBS -N stats810hw11

##This denotes the queue for the job
#PBS -q flux

##This denotes the allocation within the queue
#PBS -A stats_flux

##Setting "quality of service" = flux appears
##to be required.
#PBS -l qos=flux

##Request the number of nodes and processors
##PBS -l nodes=1:ppn=8
##For embarrassingly parallel computing, instead
##select the number of processors, sometimes
##adding one to run the master R process
#PBS -l procs=5,pmem=4000mb
```

runTest.pbs, part II

```
##This is the run time (hh:mm:ss) that your
##job will be allocated.
##It will be killed if it exceeds its walltime.
##Extreme over-estimation may slow your job
##in the queue.
#PBS -l walltime=5:00

##Import the shell’s environment
##This is important if you’re using
##Environment Modules (i.e. module load ...)
#PBS -V

##In what circumstances should an email be sent?
##’a’ is for aborted jobs,
##’b’ is when the job starts,
##’e’ is when the job exits.
#PBS -m abe

##Where should email be sent?
#PBS -M ionides@umich.edu
```

runTest.pbs, part III

```
##Concatenates standard output and error messages
##This is recommented at
##cac.engin.umich.edu/resources/software/pbs
#PBS -j oe

##code to be run

## By default,
## PBS scripts execute in your home directory,
## not the directory where they were submitted.
## The following line places you in the directory
## from which the job was submitted.
cd $PBS_O_WORKDIR

R CMD BATCH --vanilla gompertzTest.R gTest.out
```

Some quotes from Buckheit and Donoho (1995) "Wavelab and Reproducible Research" (Stanford University Statistics Department Techical Report)

Who is David Donoho?

`http://en.wikipedia.org/wiki/David_Donoho`

Burning the Midnight Oil. Once, writing an article with approximately 30 figures, we had to tweek various algorithms and display options to display clearly the effects we were looking for. As a result, after an 18-hour day we had accumulated a stack of a few hundred sheets of paper, all of which purported to be versions of the figures for the article. We gave up well after midnight. Returning to work eight hours later, we had a question: which were the final versions, the ones which should go in the article? The easy answer would have been the nicest looking ones, but that wouldn't always be right. In fact, the correct answer would have been the ones generated using the settings and algorithms exactly described in the paper. Those were not always the best-looking ones. In any event, we had a major problem sorting through the hundreds of sheets of paper to find the ones that really belonged in the article. It is possible, though not likely, that we fooled ourselves, and put the wrong version of some figures in the final copy.

Who's on First? A Graduate Student comes into a Professor's office and says, "that idea you told me to try, it doesn't work!" The Professor suggests to him some variation on the idea, and the Student returns a day later with the same response. Unfortunately, the Student's descriptions of the problems he is facing don't give the Professor much insight into what's going on. It eventually becomes apparent that the issue really is as follows: the student needs to provide the Professor with detailed information so they could explore four branches on a decision tree:

1. Is the idea itself incorrect?

2. Or is the idea okay, while the student's implementation of the idea is incorrect?

3. Or is the implementation okay, while the student's invocation of the algorithm used incorrect parameters?

4. Or is the invocation okay while the student's display of the results actually focuses on the wrong aspect of the problem?

Mere oral communications are completely inadequate to do any of this. The student has built (whether he knows that he is doing this or not) a computing environment, and unless the Professor can enter and use the Student's environment in situ as he had built it, the two couldn't possibly get a fix on the answers. But since the Student had not anticipated this issue, it was very hard for him to explain the environment (algorithms, datasets, etc.) which he had constructed, and hard for the Professor to get into it.

A Year is a Long Time in this Business. Once, about a year after one of us had done some work and written an article (and basically forgot the details of the work he had done), he had the occasion to apply the methods of the article on a newly-arrived dataset. When he went back to the old software library to try and do it, he couldn't remember how the software worked (invocation sequences, data structures, etc). In the end, he abandoned the project, saying he just didn't have time to get into it anymore.

Surely anyone reading the above recognizes the sorts of situation that we are talking about and has experienced them first-hand. It is not too much to say that these experiences are utterly common; they are the dominant experiences of researchers in those fields which rely on computational experiments. Researchers in those fields can't reproduce their own work; students in those fields can't explain to their advisers the difficulties they are having, and researchers in those fields can't reproduce the work of others. To people who have only worked in such fields, this probably seems to be just the way things are, so much so that this state of affairs is unremarkable.

That was 1995.

Solutions to these problems now exist.

Use them!

In the R/Latex community this means

knitr (a newer package upgrading Sweave)

rmarkdown

Both these packages work with RStudio (and both are associated with Yihui Xie)

rmarkdown and knitr

- Both are similar: combining text (including Latex equations) with code chunks and R output.
- **knitr** usually puts the output into a pdf format. Good for writing papers.

  ```
  <<R_code_chunk_name>>=
  my_R_function()
  @
  ```

- **rmarkdown** usually puts the output into HTML. Quicker and easier than knitr. Good for developing a research 'notebook'.

  ~~~
  ```{r R_code_chunk_name}
  my_R_function()
  ```
  ~~~

- In rmarkdown, you can use HTML and markdown as well as Latex.

Is it a problem if my code takes days to run in rmarkdown and knitr

- Results that take days to run are hard to recompute. That makes it more worthwhile to put effort into making reproducibility as easy as possible.
- However, you don't want to re-run long computations unnecessarily.
- **Caching** is carried out so that computations only get recomputed when the relevant code changes.