Computational Methods for Statistic with Applications
Computer Exercise no. 5: Sparse matrices

NGSSC, LU, SLU, UU

March 2016

The goal with this lab is that you get acquainted with some packages in $\mathbb{R}$, which enable parallel computations.
At the end, selected results of the exercises have to be sketched and sent to the lab-consultant.
Some examples of functions and script files can be downloaded via
`http://user.it.uu.se/~maya/Courses/NGSSC/index_Stat_2016.html`.
Make a copy of these files in some of your directories.

Utilizing a multicore processor

Exercise 1 ($\mathbb{R}$ package 'multicore')

Tasks

1. Start $\mathbb{R}$ and install the packages `foreach` and `doParallel`
2. Perform the following test and check the time spent for various values of n: $10^3, 10^5, 10^6, 10^7$ (`lapply.r`)

   ```
   library('doParallel')
   y <- function(x) { z <- (x^3+sqrt(x))/x^(1/3) }
   x=1:n
   system.time(lapply(x, y))

   system.time(mclapply(x, y))
   ```

   Check if you are running in parallel: in a separate terminal window type 'top' and you should see multiple tasks with your user name running simultaneously.

   Lower the computational load. Consider

   ```
   y <- function(x) { z <- (x^3}
   ```

   Check when do you have improvement in the run time (over `lapply`) when you use `mclapply`.
3. Study the examples in `foreach` and `multicore`. Observe the speedup.

4. Choose your own function and experiment. Have a look at the user manuals of `foreach`, `doParallel` - there are more possibilities to explore.

Exercise 2 Parallel computing in Matlab.
Run the example in `parfor_matlab.m` and study the following issues.

1. Does the performance improves if you have more computations?
2. For those who have access to the IT Linux servers: try the experiment with 12 or 16 workers.
3. Experience with your own code.
4. If you run Matlab on a multicore computer, does Matlab automatically take advantage of that?