Assignment 7

1. Modify your `pareto` package to include functions `p.dpareto`, `p.ppareto`, and `p.qpareto` that are implemented using C code that uses Open MP to allow the loop over elements of the argument vectors to be run in parallel. Your new functions should take one additional argument, `P`, that specifies the number of computational threads to use for the loop. Try your function on vectors of various lengths using one and two threads and try to determine how large the vector has to be for the parallel approach to improve performance. Look at the *elapsed* time values provided by `system.time`; because of the simplicity of the density you may need a fairly long vector to see the effect of parallelization.

   Some notes on Open MP are available. The notes about the use within R are out of date but the overview of Open MP is still relevant. Use of Open MP is also discussed briefly in the *Writing R Extensions* manual. A modified version of the `AddOne` package that uses Open MP is available as well. This package contains a function `p.AddOne` similar to the one you are asked to write.

   Please use the Linux systems for this as performance of Open MP on Windows and Mac OS X is not satisfactory for this problem (if it is available at all).

   Your package should pass `R CMD check` without errors or warnings. You should submit your package as a source package file created by `R CMD build`.

   Also commit and push your revised package code to your class GitLab repository using the `pareto` directory at the top level of your repository.

2. Suppose $Y_i = m(x_i) + \varepsilon_i$ with $x_i = \frac{i}{n+1}$,

$$m(x) = 1 - \sin(5x)^2 e^{-4x}$$

   and the ε_i $i.i.d$ $\mathrm{N}(0, \sigma^2)$ random variables. Consider the following estimators of m avaiable in R:

   `smooth.spline`

   `lowess`

   `supsmu`

   `loess`

   The fit produced by `gam` in packages `mgcv` for the model `y ~ s(x)`

   all using default smoothing parameter settings.

   (a) For $n = 50$ and $\sigma = 0.1$ use simulation to estimate the bias and standard error of the estimates at each x_i and show the results graphically.

(b) The average mean square error

$$aMSE(\widehat{m}) = \frac{1}{n}\sum_{i=1}^{n} E[(\widehat{m}(x_i) - m(x_i))^2]$$

is a measure of the overall quality of the estimator that approximates the integrated mean square error. Use simulation to estimate the average mean square error of the estimators for $n = 50$ and a range of σ values and summarize the results in tables and/or graphs. Comment on any performance differences you see in this example.

Make sure your simulation sample sizes are sufficient to support any conclusions.

Include your code in an appendix to your writeup and as a separate text file. Your code should make it easy to change the parameters of the simulation.

Your submission should include your source package archive for Problem 1 and a pdf file with your writeup, text files with a `.R` extension with your code for Problem 2.

You should submit your assignment electronically using Icon. Submit your work as a single compressed tar file. If your work is in a directory `mywork` then you can create a compressed tar file with the command

```
tar czf mywork.tar.gz mywork
```

Solutions and Comments

1. OpenMP is a very useful framework for taking advantage of multicore computiers. It is fairly easy to use for simple parallelizations, though some care is needed to avoid pitfalls.

   Some notes:

   - You need to make sure OpenMP is enabled; this is done by compiler and linker flags set in `Makevars`.
   - To check that your code is using multiple threads you can
     - use `htop`; for example, watch `htop` while running

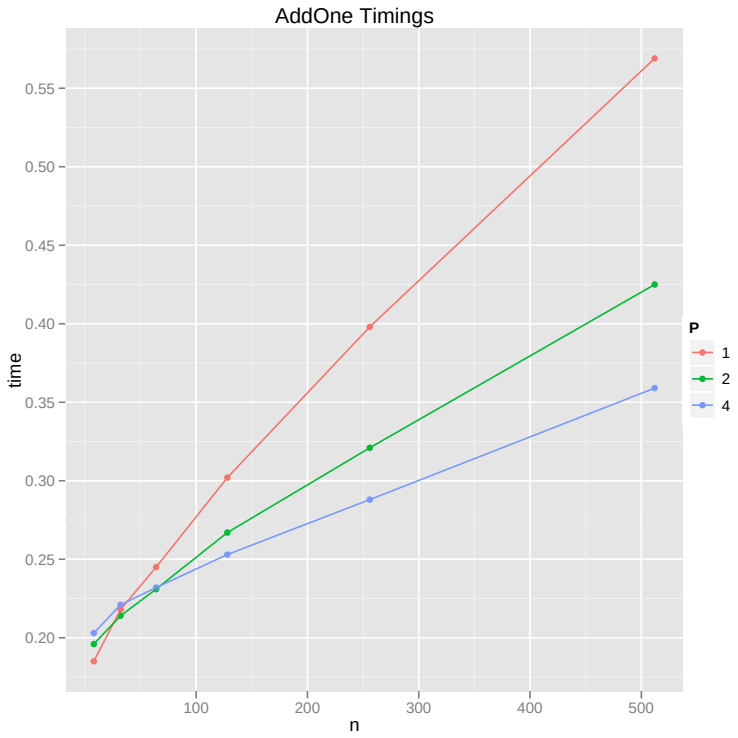
       ```
       x <-as.double(1 : 64)
       system.time(for (i in 1: 100000) p.AddOne(x, P = 4))
       ```

     - check that for large enough n your `user` time exceeds your `elapsed` time (the `user` time is cumulative over all threads in the process).
   - You need to be careful only to call functions that are *thread-safe* in the body of a parallel loop.
     - Basic arithmetic functions are safe.
     - For anything else assume it is not safe unless you have checked and made sure that it is.
     - In particular, *do not* call any functions that could allocate memory in R or signal errors or warnings.
   - Ahmdahls Law: if the proportion f of a computation is parallelizable, then using P processors will reduce computation time by a factor of

     $$1 - f + \frac{f}{P}$$

     If half your computation is in the sequential portion then you can expect at most a factor of 2 speed-up no matter how many threads you use.
     - This is an approximation.
     - The parallel and sequential components will vary approximately linarly with problem size n.
     - The sequential poriton includes both setup and memory traffic.
     - There is a cost of synchronization associated with coordinating multiple threads.
     - To make f as large as possible you should try to put all $O(n)$ operations into the parallel `for` loop.
     - This means handling recycling in C code. The C remainder operator `%` can be used for this.

- You can use *profiling* (see `?Rprof`) to see how much time your sequential code spends in the `.C` call.
- Some timing results for the `AddOne` example:



- As always, make sure to follow the coding guidelines.
- You cannot use expressions like `x == NAN` to check for a `NaN` value. Use `ISNAN` instead, if you need to.
- The `lower.tail` and `log.p` arguments should be scalars.

Some test code is available in

`http://www.stat.uiowa.edu/~luke/classes/STAT7400/paretoOMPtests.R`

2.
- Please follow the coding standards on use of spaces, indentation and long lines.
- Descriptions of simulation experiments should always include information about the simulation design, in particular the number of replicates used.
- Simulation results should include simulation standard errors.
- You need to think about the simulation standard errors to determine an appropriate simulation sample size. You may need a preliminary experiment to do this.
  - A simulation sample size of 100 is much too small.

  - Even 10,000 leaves substantial uncertainty about absolute magnitudes.
  - Some simulation variance reduction techniques can make comparisons much more accurate.
- A line plot is more effective than a point plot for showing the bias and standard error curves.
- If you use a table think about how to minimize the number of digits you need.
- Smoothness of the mean function and signal to noise ratio will affect the performance of smoothing methods.