Assignment 2

1. Redo Problem 2 from Assignment 1. Improve your definition of the `dpareto` function for computing the Pareto density; in particular make sure that the function is properly vectorized, handles the range restriction and invalid parameter values properly, and that the optional `log` argument is supported. Also make sure that your graphs properly show the discontinuity of the densities.

2. Several mechanisms are available for calling C code from R. The simplest is the `.C` interface. Modify your C function for computing the Pareto density so it can be used with the `.C` interface, and include some examples of its use in your writeup.

   The *Writing R Extensions* manual provides documentation for the `.C` interface as well as other interfaces. A simple example is provided on the class web site that you may find helpful. The files in this example show how to use the `.C` interface as well as the richer `.Call` interface.

   Your submission should include the new C source file and a source file with the R code for calling your C function. Track your work using Git and include your Git log in your submission.

Be sure to follow the coding standards. Tools are available to help:

- For C code, the `indent` program on Linux/Mac systems; these have different control options on Linux and Mac.

- For R code, the `formatR` package, in particular the `tidy_source` function in that package.

You should submit your assignment electronically using Icon. Your submission should include

- your writeup as a PDF file

- files with your R code for Problem 1 and Problem 2

- a file with your C code for Problem 2

- the Git log for your work

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

General comments:

- Don't include things not asked for (editor temporary files like `foo` , `.git` sub-directories, executables, shared libraries, etc).

- Make sure your file name ands with `.tar.gz`.

- Make sure you include a Git log in your submission when asked to.

- Avoid excessively large margins.

1.
- Please follow the coding standards on use of spaces and indentation and avoiding long lines.

- Calls to `return()` are only needed for early exit. Do not use them otherwise.

- Make sure your `log = TRUE` code works correctly.

- Use brief but informative error messages.

- It is usually numerically better to compute the log density and exponentiate than the other way around.

- One possible definition:

```
dpareto <- function(x, a, b, log = FALSE) {
    nx <- length(x)
    na <- length(a)
    nb <- length(b)
    n <- max(nx, na, nb)
    if (nx < n) x <- rep(x, length.out = n)
    if (na < n) a <- rep(a, length.out = n)
    if (nb < n) b <- rep(b, length.out = n)
    ld <- ifelse(a > 0 & b > 0,
                 ifelse(x > a,
                        log(b) + b * log(a) - (b + 1) * log(x),
                        log(0)),
                 NaN)
    if (log) ld
    else exp(ld)
}
```

This could be simpler if the `ifelse` function did not base its result size entirely on the first argument.

- These are the tests I used:

```
stopifnot(is.nan(dpareto(3,-2, 1)))  # bad parameter
stopifnot(is.nan(dpareto(3,2, -1)))  # bar parameter
stopifnot(all.equal(dpareto(3,2,1), 0.2222222222))
stopifnot(all.equal(dpareto(1,2,3), 0.0))
stopifnot(all.equal(dpareto(3:5,2, 1), c(0.2222222222, 0.1250000, 0.0800000)))
stopifnot(all.equal(dpareto(1:5,2, 1), c(0.0, 0.0, 0.2222222222, 0.1250000, 0.0800000)))
stopifnot(all.equal(dpareto(6,2:4, 1), c(0.05555555556, 0.08333333333, 0.11111111111)))
stopifnot(all.equal(log(dpareto(1:5,2, 1)), dpareto(1:5,2, 1, log = TRUE)))
stopifnot(all.equal(dpareto(1:6,1:2, 1),
                    c(0.0, 0.0, 0.11111111111, 0.125, 0.04, 0.05555555556)))
stopifnot(all.equal(dpareto(1, 2, 1:2), c(0, 0)))
```

2. • Your C code should compile without errors or warnings. You can use
      the PKG_CFLAGS environment variable to set C compiler flags to enable all
      warnings; e.g. with

      `env PKG_CFLAGS="-Wall -pedantic" R CMD SHLIB ...`

    • It is not a good idea to allocate vectors that might be large as local vari-
      ables. These will be allocated on the process stack and the space available
      for the stack is not very large.

    • Follow the coding guide on use of spaces, indentation, and long lines.

    • Standard convention in R density functions is to return NaN for bad pa-
      rameter values.

    • Do not use printf or REprintf for error messages; use the C level warn-
      ing/error calls. Do not terminate the process when there is an error.

    • It is usually numerically better to compute the log density and exponenti-
      ate than the other way around.

    • In C code you should only warn about bad parameter values once, not on
      every loop iteration.

    • You should not use loops in R if vectorized operations can be used.

    • I used the same test code as in the first problem.