Assignment 9

1. (a) Modify your `pareto` package to include a function `rpareto`, written in R only, for generating random variables from a Pareto distribution. Your function should follow the standard usage of random number generator functions in R. Your package should pass `R CMD check` without errors, warnings, or notes. In your writeup explain what method you used to implement your generator and show some usage examples.

   (b) Add a function `rcpareto` to your package that takes the same arguments as `rpareto` and generates its random numbers in C code. The section of the R extension manual on *Random number generation* provides the information you need for this. Again explain your method and show some examples of use in your writeup.

   Your package should pass `R CMD check` without errors or warnings. Your submission should include 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.

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. Some notes:

   - Your package should pass `R CMD check` without errors, warnings, or notes.
   - Follow the coding standards on indentation — make sure you do not have tabs in your code.
   - Follow the coding standards on avoiding long lines, proper indentation, and use of spaces.
   - You should use R's random number generator functions in your C code.
     - This allows the user to change the underlying generator by standard mechanisms.
     - This also allows the user to reproduce identical sequences of random numbers using the `set.seed` function.
   - You need to enclose your loop generating uniforms in calls to `GetRNGstate` and `PutRNGstate` calls.
   - You should place your `GetRNGstate/PutRNGstate` calls *outside* the loop.
   - Make sure you are generating from the correct distribution. A simple check:

     ```
     ppareto(rpareto(n, a, b), a, b)
     ```

     should look like a sample from a uniform distribution on $[0, 1]$.
   - Given that you have a working `qpareto` it makes sense to start with that for your inversion-based generator.
   - Checking generator output can be tricky:
     - Lots of statistical tests are available, but they do fail some of the time even when things are working properly.
     - Using a small α and fixing the generator and seed can help.
   - Your code should check for bad parameter values.
   - You should follow the convention that the length of `n` is used as the number of variables to generate if it is greater than 1.
   - If you are calling `warning` from your C code it it best to call `PutRNGState` first.
   - If you are using C code for generation then it makes sense to use do the inverse CDF calculation in C, not R.

   Some test code is available in

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