Assignment 3

1. Two numerical approximations to the derivative of a function f at a point x are the forward difference quotient

$$\delta_F(f, x, h) = \frac{f(x + h) - f(x)}{h}$$

and the central, or symmetric, difference quotient

$$\delta_S(f, x, h) = \frac{f(x + h) - f(x - h)}{2h}$$

for a step size h. A third option that is available when the function f is analytic near x is

$$\delta_C(f, x, h) = \frac{\Im(f(x + hi))}{h}$$

where $i = \sqrt{-1}$ is the imaginary unit and $\Im(z)$ is the imaginary part of the complex number z.

Chose a few functions and argument values and examine these approximations graphically by plotting the approximations against $-\log_2 h$ for h values in the range $2^{-1}, \dots, 2^{-64}$. Some functions and argument values you might consider:

$$\begin{aligned} f_1(x) &= \sin(x) && \text{at } x = 1 \\ f_2(x) &= 10000\sin(x) && \text{at } x = 1 \\ f_3(x) &= \tan(x) && \text{at } x = 1.59 \\ f_4(x) &= \phi(x) && \text{at } x = 0.5 \end{aligned}$$

where ϕ is the standard normal density. Comment on the behavior you see. Can you suggest a guideline for choosing the step size h?

2. Create an R package `pareto` that contains a function `dpareto` to compute the density of the Pareto distribution. Include an example in the help page and some test code in a `tests` directory. Your package should pass `R CMD check` without errors or warnings. Your writeup should contain a simple example of using your package, and you should include your package as a source package file created by `R CMD build` in your submission archive file.

The *Writing R Extensions* manual provides documentation on creating R packages. The function `package.skeleton` may help you get started. There is also a small sample package available called `AddOne` that you can start with. You can unpack the package sources with the command

```
tar zxf AddOne_1.0-1.tar.gz
```

You can find further documentation, tutorials, and tools by searching the web, e.g. for "create R package."

Commit your package source code to your UI GitHub repository in a directory named `pareto`. After your commit your repository should look like

```
<your repo>/
    README.md
    pareto/
        DESCRIPTION
        NAMESPACE
        README
        man/
            ...
        R/
            ...
        tests/
            ...
```

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

- your writeup as a PDF file
- a source code package as created by `R CMD build`.

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:

1.
    - If you use separate plots for symmetric and forward differences then you should use common axes.
    - Numerical derivatives are often used in optimization.
    - It is important to choose a step size that is not too large or too small. This balances the *truncation error* (h too large) against the *round-off error* (h too small).
    - Central differences can use larger step sizes but require more function evaluations.
    - Simple calculus can help understand why central differences will be more accurate than forward differences for a given small h value.
    - Complex differences avoid the round-off error, but require the algorithm computing the function to be able to handle complex arguments.
    - Dennis and Schnabel (1983) recommend for forward difference quotients

      $$h = \sqrt{\eta}\max\{x, t_x\}$$

      where η is the relative error in computing $f(x)$ and t_x is the typical size of x.
    - Using this rule, if $\eta = 10^D$ with D the number of accurate base 10 digits in $f(x)$ then the number of accurate digits in $\delta_F(f, x, h)$ is about $D/2$.
    - Their recommendation for central difference quotients is

      $$h = \sqrt[3]{\eta}\max\{x, t_x\}$$

      For $\eta = 10^D$ the number of accurate digits in the approximate derivative should be about $2D/3$.
    - Extrapolation methods can be useful.

2.
    - Be sure to check your code into GitHub.
    - It is usually best to only place source files under version control, not package tar balls or test results.
    - Your package should pass `R CMD check` without errors, warnings or notes.
    - It is better to explicitly export the public functions in your `NAMESPACE` file.
    - Package tests:
        - Your should include test code in a `tests` directory.
        - Your tests should try to test all important cases.

  - Be careful about floating point equality tests.

- If you include a `README` file then its contents should be appropriate for a user of your package. You can also use a `README.md` file; GitHub will render these nicely.
- Make sure your help file includes useful information.
- It is not required that `x > a`. The density is zero for `x` values outside the support; no warning is needed.
- Please follow the coding standards on use of spaces, avoiding long lines, and proper indentation.
- Vectorization should work for `x`, `a`, and `b`.
- The tests I used:

```
stopifnot(is.na(dpareto(3,-2, 1)))
stopifnot(is.na(dpareto(3,2, -1)))
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(6,1,2:4),
                    c(0.0092592593, 0.0023148148, 0.0005144033)))
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)))
```