APTS Statistical Computing Assessment 2012/13

The work provided here is intended to take up to half a week to complete. Students should talk to their supervisors to find out whether or not their department requires this work as part of any formal accreditation process (APTS itself has no resources to assess or certify students). It is anticipated that departments will decide on the appropriate *level* of assessment locally, and may choose to drop some (or indeed all) of the parts, accordingly. So make sure that your supervisor or local organizer of APTS assessment has looked at the assignment before you start, and has told you which parts of it to do. In order to avoid undermining institutions' local assessment procedures the module lecturer will not respond to enquiries from students about this assignment. Question 11 is not intended to be used for formal assessment purposes.

A number of statistical problems require estimation of covariance matrices. Sometimes it is helpful for those estimates to be robust, but several otherwise sensible robust methods result in estimates that are not necessarily positive semi definite, and may not even be symmetric. It is therefore useful to be able to find the positive semi-definite matrix, $\mathbf{B}$, that is as close as possible to any square matrix $\mathbf{C}$.

One approach is to represent $\mathbf{B}$ in terms of its Choleski factor $\mathbf{R}$, written as

$$\mathbf{R} = \begin{pmatrix} e^{\theta_1} & \theta_2 & \theta_3 & \cdot & \cdot & \theta_n \\ 0 & e^{\theta_{n+1}} & \theta_{n+2} & \cdot & \cdot & \theta_{2n-1} \\ 0 & 0 & e^{\theta_{2n}} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

where $\boldsymbol{\theta}$ is a parameter vector, and $\mathbf{B} = \mathbf{R}^\mathsf{T}\mathbf{R}$. We can then seek $\hat{\boldsymbol{\theta}}$ to minimise

$$f(\theta) = \sum_{ij} |B_{ij} - C_{ij}|^\alpha$$

where α is some fixed constant greater than 1.

Here is some R code to simulate a suitable example matrix $\mathbf{C}$

```
n <- 10;set.seed(1)
C <- matrix(runif(n^2),n,n)
C <- C + t(C) + matrix(rnorm(n^2)*.3,n,n)
```

Answer the following questions. When writing R code, your functions should all be general enough to deal with a square matrix of any reasonable dimension (i.e. not just 10×10).

1. Explain why the representation of $\mathbf{B}$ is a sensible way of parameterizing a positive (semi) definite matrix.

2. Write an R routine to take a parameter vector $\boldsymbol{\theta}$ (`theta`) and convert it into the equivalent Choleski factor $\mathbf{R}$.

3. Write an R routine to take a Choleski factor $\mathbf{R}$ and turn it into a parameter vector $\boldsymbol{\theta}$ (`theta`).

4. Write an R function which takes arguments `theta` ($\boldsymbol{\theta}$), `C` and `alpha` (α), and evaluates f.

5. Write an R function (same arguments as previous part) to evaluate the vector of $\partial f/\partial \theta_i$ values. Do take care to carefully work out the mathematical expressions for the required derivatives before you try coding anything. Also take care to avoid wasteful computation such as element wise multiplication of 2 matrices where one of the matrices is all zeroes except for a single row or column! It will be hard to avoid some looping in this function, but you should require at most 2 levels of nested loop.

6. Write R code to test your routine from part 5 by finite differencing. If there is a problem, go back to 5 and fix it!

7. Propose a method for finding initial guesstimates of $\boldsymbol{\theta}$ from $\mathbf{C}$, and code it up (Hint: eigen and Choleski decompositions are useful here).

8. Now use your objective and gradient function from parts 4 and 5 to find $\mathbf{B}$ for $\alpha = 1.3$, using the BFGS method in R function `optim`. Check the `optim` help file to see how to supply gradients to `optim`, and also how to increase the optimizer iteration limit (to at least 200). Sanity check your result! (Hint: your optimized objective function value should be about 23.56).

9. Use `system.time` to compare `optim`'s speed for BFGS with and without supplied gradients. If you have coded efficiently, then you should find a 10 fold speed up with supplied gradients (rather than the FD gradients that `optim` uses otherwise).

10. Use your code to confirm that when $\alpha = 2$ then $\mathbf{B} = \mathbf{U}\boldsymbol{\Lambda}^{+}\mathbf{U}^{\mathsf{T}}$ where $(\mathbf{C} + \mathbf{C}^{\mathsf{T}})/2 = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}}$ is an eigen decomposition and $\boldsymbol{\Lambda}^{+}$ is $\boldsymbol{\Lambda}$ with all the negative entries set to zero.

11. For a bonus mark, prove the result from 10, for the case where $\mathbf{C}$ is symmetric.