APTS Statistical Computing
Assessment 2018/19

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. 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 organiser 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.

The assignment is to develop an R function for efficiently and stably fitting a generalised linear model (GLM) with a one-parameter exponential family observation model. We will first derive an appropriate algorithm, then implement it, and test it for the important special case of logistic regression.

A one-parameter exponential family model with canonical parameter θ has a density (or mass function) of the form

$$f(y|\theta) = \exp\{\theta y - b(\theta) + c(y)\},$$

for some scalar functions $b(\cdot)$ and $c(\cdot)$. In a GLM, a linear predictor $\mathbf{X}\boldsymbol{\beta}$ is used to model the canonical parameter θ, for p-dimensional regression parameters $\boldsymbol{\beta}$ and an $n \times p$ design matrix $\mathbf{X}$ (here assumed fixed and known). Assume an n-dimensional vector of observations $\mathbf{y} = (y_1, y_2, \ldots, y_n)$.

1. Show that the log-likelihood for this model (up to a constant) can be written in the form

   $$l(\boldsymbol{\beta}; \mathbf{y}) = \mathbf{y}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} - \mathbf{1}^\mathsf{T} b(\mathbf{X}\boldsymbol{\beta}),$$

   where $\mathbf{1}$ is an n-dimensional vector of 1s, $^\mathsf{T}$ denotes transpose, and the function $b(\cdot)$ is "vectorised".

2. Derive the gradient, $\nabla l(\boldsymbol{\beta})$, and Hessian matrix, $\nabla^2 l(\boldsymbol{\beta})$.

3. Use the gradients to derive a Newton–Raphson updating scheme for maximising this likelihood wrt $\boldsymbol{\beta}$, and show that the updates can be written in the form

   $$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + [\mathbf{X}^\mathsf{T} \operatorname{diag}\{\mathbf{w}_k\}\mathbf{X}]^{-1}\mathbf{X}^\mathsf{T}\mathbf{z}_k,$$

   where $\mathbf{w}_k = b''(\mathbf{X}\boldsymbol{\beta}_k)$ and $\mathbf{z_k} = \mathbf{y} - b'(\mathbf{X}\boldsymbol{\beta}_k)$.

4. By defining $\mathbf{X}_k = \operatorname{diag}\{\mathbf{w}_k\}^{1/2}\mathbf{X}$, show that the N–R update can alternatively be written implicitly in the form

   $$\mathbf{R}_k(\boldsymbol{\beta}_{k+1} - \boldsymbol{\beta}_k) = \mathbf{Q}_k{}^\mathsf{T}(\mathbf{w}_k^{-1/2} \circ \mathbf{z}_k),$$

   where $\mathbf{X}_k = \mathbf{Q}_k\mathbf{R}_k$ is the QR decomposition of $\mathbf{X}_k$ and $\circ$ is the Hadamard (elementwise) product.

5. Write a function, `glm1`, to fit a one-parameter GLM to data using the above N–R scheme. It should have as input parameters, `y`, the data, `X`, the design matrix, `bp`, the function $b'(\cdot)$ (assumed vectorised) and `bpp`, the function $b''(\cdot)$ (also vectorised). The output should be the vector of regression coefficients, $\boldsymbol{\beta}$, at convergence.

   - Your function should *not* call either of the built-in R functions `lm`, `glm`, or similar.

- It should also *not* use the function `solve`, either for explicit matrix inversion, or general solution of an arbitrary linear system. You *may* use a substitution solver such as `forwardsolve` or `backsolve`, as appropriate.
- Your function should *not* create or use any $n \times n$ matrices (even diagonal ones), should not explicitly invert any matrix, and should not explicitly create matrices of the form $\mathbf{X}^\mathsf{T}\mathbf{X}$ or $\mathbf{X}^\mathsf{T}\mathbf{D}\mathbf{X}$ (for diagonal $\mathbf{D}$).
- You will need to use an appropriate termination criterion for your iteration scheme.

6. Show that a Bernoulli observation model

$$f(y|p) = p^y(1-p)^{1-y}, \quad y \in \{0, 1\},$$

can be written in one-parameter exponential form with canonical parameter $\theta = \log(p/(1-p))$. Identify the function $b(\cdot)$, and compute $b'(\cdot)$ and $b''(\cdot)$.

7. A GLM with a Bernoulli observation model and canonical link is often known as a *logistic regression* model. We are now ready to write a function to fit a logistic regression model, exploiting the function `glm1`. We can use R's model formula parsing functions to allow it to use friendly syntax as follows.

```
logReg <- function(formula, data)
{
  mf=model.frame(formula, data = data)
  y=model.response(mf)
  if (is.factor(y))
      y = as.numeric(y) - 1
  X=model.matrix(formula, mf)
  bp = function(th) {
      e = exp(-th)
      1/(e+1)
  }
  bpp = function(th) {
      e = exp(-th)
      e/((e+1)^2)
  }
  glm1(y, X, bp, bpp)
}
```

We can now test this with an appropriate real or synthetic dataset and cross-check our fitted regression coefficients computed by R's built-in `glm` function. For example, if you have the CRAN `MASS` package installed, we can fit the Pima Indians dataset with:

```
logReg(type ~ ., data = MASS::Pima.tr)
```

and compare results against `glm`'s with:

```
glm(type ~ ., data = MASS::Pima.tr, family = "binomial")
```

Your fitted coefficients should match those of the `glm` function to several decimal places.