# Solutions to APTS Statistical Computing Assessment 2022/23

André F. B. Menezes, Maynooth University

January 27, 2023

## 1 Introduction

This report contains my solutions for the APTS Statistical Computing Assessment from 2022/23. 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. Further details can be found here.

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

$$f(y \mid \theta) = \exp\left\{\theta y - b(\theta) + c(y)\right\}$$

where $b(\cdot)$ and $c(\cdot)$ are known scalar functions.

In a GLM the linear predictor $\eta = \mathbf{X}\beta$ is related to the model canonical parameter, $\theta$, thought a convenience link function, where $\beta$ is a $p$-dimensional regression parameters and $\mathbf{X}$ is an $n \times p$ design matrix with known values. Moreover, we will denote the $\mathbf{y} = (y_1, \dots, y_n)$ as the $n$-dimensional vector of observations.

## 2 Developed solution

First, we need to write the log-likelihood function in matrix form. Let be $\theta_i = \mathbf{x}_{i\bullet}^\top \beta$ and $\mathbf{x}_{i\bullet} = (x_{i0}, x_{i1}, \dots, x_{i(p-1)})$, then the log-likelihood for $\beta = (\beta_0, \beta_1, \dots, \beta_{p-1})$, up to a constant, is given by

$$\begin{aligned}\ell(\beta) &= \sum_{i=1}^{n} \theta_i \, y_i - \sum_{i=1}^{n} b(\theta_i) \\ &= \sum_{i=1}^{n} \mathbf{x}_{i\bullet}^{\top}\beta \, y_i - \sum_{i=1}^{n} b(\mathbf{x}_{i\bullet}^{\top}\beta).\end{aligned}$$

We can use the summing vector, $\mathbf{1}^{\top} = (1, \dots, 1)$ to express a sum of a number in matrix notation, hence the log-likelihood can be re-written as follows

$$\ell(\beta) = \mathbf{y}^{\top}\, \mathbf{X}\beta - \mathbf{1}^{\top}\, b(\mathbf{X}\beta)$$

where now $b(\cdot)$ is a $p$-dimensional vectorised function.

To obtain the gradient and Hessian matrix we first, need to note that the log-likelihood can be written as

$$\begin{aligned}\sum_{i=1}^{n} \mathbf{x}_{i\bullet}^{\top}\beta \, y_i &= \mathbf{x}_{1\bullet}^{\top}\beta \, y_1 + \dots + \mathbf{x}_{n\bullet}^{\top}\beta \, y_n \\ &= (x_{10}, x_{11}, \dots, x_{1(p-1)})\,(\beta_0, \dots, \beta_{p-1})\, y_1 + \dots + (x_{n0}, x_{n1}, \dots, x_{n(p-1)})\,(\beta_0, \dots, \beta_{p-1})\, y_n\end{aligned}$$

Then, it's straightforward to derive the derivative for $\beta_k$ as

$$\begin{aligned}\frac{\partial}{\partial \beta_k}\ell(\beta) &= \sum_{i=1}^{n} x_{ik}\, y_i - \sum_{i=1}^{n} b'(\mathbf{x}_{i\bullet}^{\top}\beta)\, x_{ik} \\ &= \sum_{i=1}^{n} \left[y_i - b'(\mathbf{x}_{i\bullet}^{\top}\beta\right]\, x_{ik}\end{aligned}$$

for $k = 1, \dots, p$

Then, the score vector can be defined as follows:

$$\Delta\ell(\beta) = \frac{\partial}{\partial\beta}\,\ell(\beta) = \mathbf{X}^{\top}\left[\mathbf{y} - b'(\mathbf{X}\beta)\right]$$

Concerning the Hessian matrix, note that the diagonal elements are given by:

$$\frac{\partial^2}{\partial\beta_k^2}\,\ell(\beta) = -\sum_{i=1}^{n} b''(\mathbf{x}_{i\bullet}^{\top}\beta)\, x_{ik}^2, \quad k = 1, \dots, p.$$

And the off-diagonal terms are all zero, i.e.,

$$\frac{\partial^2}{\partial \beta_j \beta_k^2} \ell(\beta) = 0, \quad j \neq k, \quad j, k = 1, \dots, p.$$

Therefore, the Hessian matrix can be expressed as

$$\Delta^2 \ell(\beta) = \mathbf{H}(\beta) = -\mathbf{X}^\top \operatorname{diag}(\omega)\, \mathbf{X}$$

where, $\omega = b''(\mathbf{X}\beta)$.

Given a initial parameter guess $\beta^{(0)} = (\beta_0^{(0)}, \beta_1^{(0)}, \dots, \beta_{p-1}^{(0)})$ the Newton-Raphson (NR) update schema is given by

$$\beta^{(k+1)} = \beta^{(k)} - \mathbf{H}^{-1}\left(\beta^{(k)}\right) \Delta(\beta^{(k)}).$$

For the one parameter exponential family model we can write the schema as

$$\begin{aligned}
\beta^{(k+1)} &= \beta^{(k)} + \left[\mathbf{X}^\top \operatorname{diag}(\omega_k)\, \mathbf{X}\right]^{-1} \mathbf{X}^\top \left[\mathbf{y} - b'(\mathbf{X}\,\beta_k)\right] \\
&= \beta^{(k)} + \left[\mathbf{X}^\top \operatorname{diag}(\mathbf{w}_k)\, \mathbf{X}\right]^{-1} \mathbf{X}^\top \mathbf{z}_k
\end{aligned}$$

where $\mathbf{w}_k = b''(\mathbf{X}\,\beta_k)$ and $\mathbf{z}_k = \mathbf{y} - b'(\mathbf{X}\,\beta_k)$.

We can write the NR update in a more efficient computational form by the defining $\mathbf{X}_k = \operatorname{diag}\{\mathbf{w}_k\}^{1/2}\, \mathbf{X}$. Consider the QR decomposition of $\mathbf{X}_k = \mathbf{Q}_k\, \mathbf{R}_k$, such that $\mathbf{R}_k$ is an upper triangular $n \times p$ matrix and $\mathbf{Q}_k$ has the same dimension of X_k with orthogonal columns, so $\mathbf{Q}_k^\top\, \mathbf{Q}_k = \mathbf{I}_{n \times p}$. Also, note that $\operatorname{diag}\{\mathbf{w}_k\} = \operatorname{diag}\{\mathbf{w}_k\}^{1/2} \operatorname{diag}\{\mathbf{w}_k\}^{1/2}$, then the NR update can be set up as:

$$\begin{aligned}
\beta^{(k+1)} &= \beta^{(k)} + \left[\mathbf{X}^\top \operatorname{diag}(\mathbf{w}_k)^{1/2} \operatorname{diag}(\mathbf{w}_k)^{1/2}\, \mathbf{X}\right]^{-1} \mathbf{X}^\top \mathbf{z}_k \\
&= \beta^{(k)} + \left[\mathbf{X}^\top \operatorname{diag}(\mathbf{w}_k)^{1/2}\, \mathbf{X}_k\right]^{-1} \mathbf{X}^\top \mathbf{z}_k \\
&= \beta^{(k)} + \left[\mathbf{X}^\top \operatorname{diag}(\mathbf{w}_k)^{1/2}\, \mathbf{Q}_k\, \mathbf{R}_k\right]^{-1} \mathbf{X}^\top \mathbf{z}_k \\
&= \beta^{(k)} + \mathbf{R}_k^{-1}\, \mathbf{Q}_k^{-1} \left[\operatorname{diag}(\mathbf{w}_k)\right]^{-1/2} \left(\mathbf{X}^\top\right)^{-1} \mathbf{X}^\top \mathbf{z}_k \\
&= \beta^{(k)} + \mathbf{R}_k^{-1}\, \mathbf{Q}_k^\top \left[\operatorname{diag}(\mathbf{w}_k)\right]^{-1/2} \mathbf{z}_k.
\end{aligned}$$

Since $\operatorname{diag}(\mathbf{w}_k)$ is a $n \times p$ diagonal matrix and $\mathbf{z}_k$ is a $p \times 1$ vector we can write the matrix multiplication as a Hadamard (elementwise) product, which is more computational efficient. It turns out that

$$\begin{aligned}
\beta^{(k+1)} &= \beta^{(k)} + \mathbf{R}_k^{-1}\, \mathbf{Q}_k^\top \left[\mathbf{w}_k^{-1/2} \circ \mathbf{z}_k\right] \\
\beta^{(k+1)} - \beta^{(k)} &= \mathbf{R}_k^{-1}\, \mathbf{Q}_k^\top \left[\mathbf{w}_k^{-1/2} \circ \mathbf{z}_k\right] \\
\mathbf{R}_k\,(\beta^{(k+1)} - \beta^{(k)}) &= \mathbf{Q}_k^\top \left[\mathbf{w}_k^{-1/2} \circ \mathbf{z}_k\right].
\end{aligned}$$

The following function glm1 gives an implementation of the NR schema to estimate the parameters of a one-parameter exponential family generalised linear model. The function does not perform matrix inversion. Note that the left hand side of NR update is upper triangular, because of the $\mathbf{R}_k$ matrix, then the backsolve function can be use in order to efficient solve a triangular system of linear equations. The termination criterion is defined as $\max\left|\beta^{(k)} - \beta^{(k+1)}\right| < \epsilon$, where $\epsilon = 10^{-6}$. The function returns a data.frame with the estimates parameter for each iteration until reach the termination criterion.

```r
glm1 <- function(y, X, bp, bpp) {

  p <- NCOL(X)
  beta_k <- rep(0, p)
  list_beta <- list()
  stop_error <- 1e-6
  j <- 1L
  current_error <- 1

  while (current_error > stop_error) {
    eta_k <- X %*% beta_k
    z_k <- y - bp(eta_k)
    w_k <- bpp(eta_k)
    X_k <- drop(w_k^(1/2)) * X
    wz_k <- w_k^(-1/2) * z_k
    qr_out <- qr(X_k)
    Q_k <- qr.Q(qr_out)
    R_k <- qr.R(qr_out)
    a_k <- backsolve(R_k, crossprod(Q_k, wz_k))

    # Update the parameter
    list_beta[[j]] <- a_k + beta_k
    current_error <- max(abs(beta_k - list_beta[[j]]))
    beta_k <- list_beta[[j]]
    j <- j + 1L
  }

  do.call(cbind, list_beta)
}
```

Regarding the Bernoulli GLM with probability mass function given by

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

Its p.m.f. can be written in the one-parameter exponential family as follows

$$\begin{aligned} f(y \mid p) &= \exp\{y \log p + (1-y) \log(1-p)\} \\ &= \exp\{y \log p - y \log(1-p) + \log(1-p)\} \\ &= \exp\left\{y \log\left(\frac{p}{1-p}\right) + \log(1-p)\right\}. \end{aligned}$$

By defining $\theta = \log\left(\frac{p}{1-p}\right)$ it turns out that

$$e^\theta = \frac{p}{1-p} \rightarrow e^\theta - p\, e^\theta = p \rightarrow p = \frac{e^\theta}{1+e^\theta}$$

and

$$1-p = 1 - \frac{e^\theta}{1+e^\theta} = \frac{1+e^\theta-e^\theta}{1+e^\theta} = \frac{1}{1+e^\theta}.$$

Hence,

$$\begin{aligned} f(y \mid p) &= \exp\left\{y\,\theta + \log\left[\frac{1}{1+e^\theta}\right]\right\} \\ &= \exp\left\{y\,\theta - \log\left(1+e^\theta\right)\right\} \end{aligned}$$

where $b(\theta) = \log\left(1+e^\theta\right)$.

Therefore,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\,\theta} b(\theta) &= \frac{e^\theta}{1+e^\theta} = \frac{1}{1+e^{-\theta}} \\ \frac{\mathrm{d}^2}{\mathrm{d}\,\theta^2} b(\theta) &= (-1)\left(-e^{-\theta}\right)\left(1+e^{-\theta}\right)^{-2} \frac{e^\theta}{1+e^\theta} = \frac{e^{-\theta}}{(1+e^{-\theta})^2}. \end{aligned}$$

Finally, a user friendly function, `logReg`, which call the `glm1` is built to fit a logistic regression model.

```r
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(theta) 1 / (1 + exp(-theta))
  bpp <- function(theta) {
    e <- exp(-theta)
```

```r
      e / (1 + e)^2
    }
    glm1(y, X, bp, bpp)
  }
```

In order to cross-check the estimated regression coefficients computed by `glm1` and the R's built-in `glm` function two examples are provided. First, I simulated 500 samples from a logistic regression model with two covariates, where the true values of the coefficients are $\beta = (1.0, -0.5, 0.5)$ and the covariates were simulated from standard Normal and uniform random variables as follows.

```r
set.seed(69)
n <- 500
X <- cbind(1, rnorm(n), runif(n))
betas <- c(1, -0.5, 0.5)
eta <- drop(X %*% betas)
p <- 1 / (1 + exp(-eta))
y <- rbinom(n, size = 1, prob = p)
sim_data <- data.frame(y = y, x1 = X[, 2], x2 = X[, 3])
```

The estimated coefficients from `logReg` and `glm` are identical at seven decimal places.

```r
out_logReg <- logReg(formula = y ~ x1 + x2, data = sim_data)
out_glm <- glm(formula = y ~ x1 + x2, data = sim_data, family = "binomial")
cbind(true = betas,
      logReg = out_logReg[, ncol(out_logReg)],
      glm = coef(out_glm))
```

```
            true     logReg        glm
(Intercept)  1.0  1.0698941  1.0698941
x1          -0.5 -0.5386558 -0.5386558
x2           0.5  0.5473424  0.5473424
```

The second cross-checks uses the data `Pima.tr` from `MASS` package. Again, the estimated values are identical at nine decimal places for all parameters. Therefore, those examples showed that the `glm1` function is successful implemented.

```r
out_logReg <- logReg(type ~ ., data = MASS::Pima.tr)
out_glm <- glm(type ~ ., data = MASS::Pima.tr, family = "binomial")
```

```r
cbind(logReg = out_logReg[, ncol(out_logReg)],
      glm = coef(out_glm))
```

```
                  logReg          glm
(Intercept) -9.773061533 -9.773061533
npreg        0.103183427  0.103183427
glu          0.032116823  0.032116823
bp          -0.004767542 -0.004767542
skin        -0.001916632 -0.001916632
bmi          0.083623912  0.083623912
ped          1.820410367  1.820410367
age          0.041183529  0.041183529
```