Introduction to the R Statistical Computing Environment
R Programming 2: Exercises

John Fox
(McMaster University)
ICPSR

2018

1. * *A straightforward problem:* Write an R function for linear least-squares regression. You might call the function `ls`, with arguments `X`, for the model matrix, and y for the response vector. Optionally include an argument `intercept`, defaulting to `TRUE`, controlling whether a column of ones should be added to the model matrix (but do, in any event, make provision for an intercept. If you know how to do least-squares regression by a QR or singular-value decomposition, then by all means feel free to do that, but I suggest that you simply compute the least-squares solution $\mathbf{b}$, fitted values $\widehat{\mathbf{y}}$, residuals $\mathbf{e}$, and the estimated covariance matrix of the coefficients $\widehat{V}(\mathbf{b})$ as

$$\begin{aligned}
\mathbf{b} &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \\
\widehat{\mathbf{y}} &= \mathbf{X}\mathbf{b} \\
\mathbf{e} &= \mathbf{y}-\widehat{\mathbf{y}} \\
\widehat{V}(\mathbf{b}) &= \frac{\mathbf{e}'\mathbf{e}}{n-p}(\mathbf{X}'\mathbf{X})^{-1}
\end{aligned}$$

   where n is the number of rows (observations) of $\mathbf{X}$ and p is the number of columns (parameters). Have your function return a list with the coefficients (say, `coef`), fitted values (`fitted`), residuals (`residuals`), and coefficient covariance matrix (`vcov`).

   Test your function with the `longley` data set included with R. This data set was used by W. Longley (1967), "An appraisal of least-squares programs from the point of view of the user," *Journal of the American Statistical Association*, **62**, 819–841, to demonstrate the instability of then-current statistical software with highly collinear data. See `?longley` for details of the data. Using both your ls function and the standard R `lm` function, compute the least-squares regression coefficients; for example

```
mod.lm <- lm(Employed ~., data=longley)
coef(mod.lm)

X <- data.matrix(longley[, 1:6])
y <- longley[, "Employed"]
mod.ls <- ls(X, y)
mod.ls$coef

mod.ls$coef/coef(mod)
```

2. *A challenging problem:* Write a binary logistic-regression function that works by using `optim()` to maximize the log-likelihood

$$\log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} y_i \log \pi_i + (1 - y_i)(1 - \pi_i)$$

where $y_i = 0$ for a "failure" or 1 for a "success" for the ith of n cases; and

$$\pi_i = \frac{1}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta})}$$

is the probability of success for the ith case. Here, $\mathbf{x}_i'$ is the vector of regressors for the ith case (i.e., the ith row of the model matrix $\mathbf{X}$), and $\boldsymbol{\beta}$ is the vector of logistic-regression coefficients, which we want to estimate. The gradient of the log-likelihood for this model is

$$\frac{\partial \log L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} (y_i - \pi_i)\mathbf{x}_i$$

Your function should take the response vector `y` and model matrix `X` as inputs, and return a list with the estimated regression coefficients and their estimated asymptotic covariance matrix (which you can compute from the Hessian returned by `optim()`).

3. *Another challenging problem:* Another way of finding the MLE in binary logistic regression is to use the Newton-Raphson algorithm, which works as follows:

   (a) Choose initial estimates of the regression coefficients, such as $\boldsymbol{\beta}_0 = \mathbf{0}$ (i.e., a vector of zeroes).

   (b) At each iteration t update the regression coefficients:

   $$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} + (\mathbf{X}'\mathbf{V}_{t-1}\mathbf{X})^{-1}\mathbf{X}'(\mathbf{y} - \boldsymbol{\pi}_{t-1})$$

   where $\mathbf{X}$ is the model matrix with $\mathbf{x}_i'$ as its ith row; $\mathbf{y}$ is the response vector of 0s and 1s; $\boldsymbol{\pi}_{t-1}$ is the vector of fitted response probabilities from the previous iteration, with ith element

   $$\pi_{i,t-1} = \frac{1}{1 + \exp(-\mathbf{x}_i'\boldsymbol{\beta}_{t-1})}$$

   and $\mathbf{V}_{t-1}$ is a diagonal matrix with diagonal elements $\pi_{i,t-1}(1 - \pi_{i,t-1})$.

   (c) Repeat step (a) until $\boldsymbol{\beta}_t$ is close enough to $\boldsymbol{\beta}_{t-1}$, at which point $\boldsymbol{\beta}_t$ is the MLE $\widehat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ (to a close approximation). The estimated asymptotic covariance matrix of the coefficients is then given by $(\mathbf{X}'\mathbf{V}\mathbf{X})^{-1}$ from the last iteration.

   Program this method as an R function, taking the response vector `y` and model matrix `X` as inputs, and return a list with the estimated regression coefficients and their covariance matrix.

4. *Yet another challenging problem:* Iterated weighted least squares (IWLS) is a standard method of fitting generalized linear models to data. As described in Section 5.12 of the *R Companion*, the IWLS algorithm applied to *binomial* (i.e., not necessarily *binary*) logistic regression proceeds as follows:

(a) Set the regression coefficients to initial values, such as $\boldsymbol{\beta}^{(0)} = \mathbf{0}$ (where the superscript 0 indicates start values).

(b) At each iteration t calculate the current fitted probabilities $\boldsymbol{\mu}$, variance-function values $\boldsymbol{\nu}$, working-response values $\mathbf{z}$, and weights $\mathbf{w}$:

$$\begin{aligned}
\mu_i^{(t)} &= [1 + \exp(-\eta_i^{(t)})]^{-1} \\
v_i^{(t)} &= \mu_i^{(t)}(1 - \mu_i^{(t)}) \\
z_i^{(t)} &= \eta_i^{(t)} + (y_i - \mu_i^{(t)})/v_i^{(t)} \\
w_i^{(t)} &= n_i v_i
\end{aligned}$$

Here, n_i represents the binomial denominator for the ith observation; for binary data, all of the n_i are 1.

(c) Regress the working response on the predictors by weighted least squares, minimizing the weighted residual sum of squares

$$\sum_{i=1}^{n} w_i^{(t)}(z_i^{(t)} - \mathbf{x}_i'\boldsymbol{\beta})^2$$

where $\mathbf{x}_i'$ is the ith row of the model matrix.

(d) Repeat steps (b) and (c) until the regression coefficients stabilize at the maximum-likelihood estimator $\widehat{\boldsymbol{\beta}}$.

(e) Calculate the estimated asymptotic covariance matrix of the coefficients as

$$\widehat{\mathcal{V}}(\widehat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}$$

where $\mathbf{W} = \mathrm{diag}\{w_i\}$ is the diagonal matrix of weights from the last iteration and $\mathbf{X}$ is the model matrix.

Problem: Program this method in R. The function that you define should take (at least) three arguments: The model matrix `X`; the response vector of observed proportions `y`; and the vector of binomial denominators `n`. I suggest that you let `n` default to a vector of `1`s (i.e., for binary data, where `y` consists of `0`s and `1`s), and that you attach a column of `1`s to the model matrix for the regression constant so that the user does not have to do this when the function is called.

Programming hints:

- Adapt the structure of the example developed in Section 8.5.1 of the *R Companion* (but note that the example is for *binary* logistic regression, while the current exercise is to program the more general *binomial* logit model).
- Use the `lsfit()` function to get the weighted-least-squares fit, calling the function as `lsfit(X, z, w, intercept=FALSE)`, where `X` is the model matrix; `z` is the current working response; and `w` is the current weight vector. The argument `intercept=FALSE` is needed because the model matrix already has a column of `1`s. The function `lsfit` returns a list, with element `$coef` containing the regression coefficients. See `help("lsfit")` for details.

- One tricky point is that `lsfit()` requires that the weights (`w`) be a *vector*, while your calculation will probably produce a *one-column matrix* of weights. You can coerce the weights to a vector using the function `as.vector()`.
- Return a list with the maximum-likelihood estimates of the coefficients, the covariance matrix of the coefficients, and the number of iterations required.
- You can test your function on the `Mroz` data in the **car** package, being careful to make all of the variables numeric (as in Section 8.5.1). You might also try fitting a binomial (as opposed to binary) logit model.

5. *Maybe the best problem:* Pick a statistical method with which you are intimately familiar and program it in R.

6. *Object-oriented programming:* Convert your linear least-squares function from Exercise 1 or your logistic-regression function from Exercise 2, 3, or 4 into an S3 generic function with `default` and `formula` methods. Write `print()` and `summary()` methods for objects created by your function. Also write methods for some of the standard R modeling generic functions (unless there's a `default` method that works) — `vcov()` (covariance matrix of coefficients), `coef()` (coefficient vector), `fitted()` (fitted values), `residuals()`, and so on.