

Part 6, Debugging, Object-Oriented Programming: Exercises

Introduction to R
McMaster

Fall 2005

1. *Debugging Functions*: Here are “solutions” to the preceding three programming problems, but each function has a bug (or two) that either causes it to fail or, possibly only in certain circumstances, to give the wrong answer. In each case, find the bug(s) and fix the function. A file with the bugged functions is available for download from the course web site.

- a. A function to calculate logistic-regression estimates by iteratively reweighted least-squares:

```
lregIWLs <- function(X, y, n=rep(1,length(y)), maxIter=10, tol=1E-6){ # bugged!
  # X is the model matrix
  # y is the response vector of observed proportion
  # n is the vector of binomial counts
  # maxIter is the maximum number of iterations
  # tol is a convergence criterion
  X <- cbind(1, X) # add constant
  b <- bLast <- rep(0, ncol(X)) # initialize
  it <- 1 # iteration index
  while (it <= maxIter){
    if (max(abs(b - bLast)/(abs(bLast) + 0.01*tol)) < tol)
      break
    eta <- X %*% b
    mu <- 1/(1 + exp(-eta))
    nu <- as.vector(mu*(1 - mu))
    w <- n*nu
    z <- eta + (y - mu)/nu
    b <- lsfit(X, z, w, intercept=FALSE)$coef
    bLast <- b
    it <- it + 1 # increment index
  }
  if (it > maxIter) warning('maximum iterations exceeded')
  Vb <- solve(t(X) %*% diag(w) %*% X)
  list(coefficients=b, var=Vb, iterations=it)
}
```

b. A function to compute the row-echelon form of a matrix by Gaussian elimination:

```
rowEchelonForm <- function(A){ # bugged!
  n <- nrow(A)
  m <- ncol(A)
  for (i in 1:min(c(m, n))){
    currentColumn <- A[,i]
    currentColumn[1:n < i] <- 0
    which <- which.max(abs(currentColumn)) # find maximum pivot in current
                                           # column at or below current row

    pivot <- A[which, i]
    if (abs(pivot) == 0) next # check for 0 pivot
    if (which > i) A[c(i, which),] <- A[c(which, i),] # exchange rows
    A[i,] <- A[i,]/pivot # pivot
    row <- A[i,]
    A <- A - outer(A[,i], row) # sweep
    A[i,] <- row # restore current row
  }
  for (i in 1:n) if (max(abs(A[i,1:m])) == 0)
    A[c(i,n),] <- A[c(n,i),] # 0 rows to bottom
  A
}
```

Note that this function returns the right answer for the matrix used as an example in Problem 2,

$$\begin{bmatrix} -2 & 0 & -1 & 2 \\ 4 & 0 & 1 & 0 \\ 6 & 0 & 1 & 2 \end{bmatrix}$$

whose row-echelon form is

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

but gives the wrong answer for the matrix

$$\begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

whose correct row-echelon form is

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

c. A function to calculate running medians:

```
runningMedian <- function(x, length=3){ # bugged!
#   x: a numeric vector
#   length: the number of values for each running median, defaults to 3
  n <- length(x)
  X <- matrix(x, n, length)
  for (i in 1:length) X[1:(n - i + 1), i] <- x[-(1:(i - 1))]
  apply(X, 1, median)[1:(n - length + 1)]
}
```

2. *Object-Oriented Programming*: Recall the function `lreg.3`, which returns objects of class `lreg`.

- a. Write methods for the generic functions `coef`, `vcov`, and `deviance`, to return the logistic-regression coefficients, their covariance matrix, and the residual deviance for objects of this class.
- b. More ambitiously, write a method for the generic function `anova` to compare two objects of class `lreg` by a likelihood-ratio test, supposing that one object represents a model that is properly nested within the other. Allow the larger model to be given either first or second, and try to check that the models are in fact nested.