「R」におけるGWRモデルの係数推定法

大下　祐樹

2008年1月28日

1　GWRモデル

地理的加重回帰モデル(以後GWRモデル)とは、xとyの関係自体が空間的に変動していると考えたモデルであり回帰係数β_0,β_1が地区ごとに変動する。
地区iのパラメータβ_{0i},β_{1i}を推定するとき

$$\mathbf{W_i}y = \mathbf{W_i}\mathbf{X}\boldsymbol{\beta_i} + \boldsymbol{\epsilon} \tag{1}$$

ここに

$$\begin{aligned} \mathbf{W_i} &= \mathrm{diag}\,(h_{i1}, h_{i2}, \cdots, h_{ij}) \\ \boldsymbol{\beta_i} &= (\beta_{0i}\ \ \beta_{1i})^T \end{aligned}$$

パラメータβ_iは重み付け最小二乗法を解くことによって求める。地区iを推定する際の重み付け最小二乗法は

$$\min[\sum_{j=1}^{n}\{y_j - (\beta_{0i} + \beta_{1i}x_j)\}^2 h_{ij}^2]$$

となり、推定パラメータは

$$\hat{\beta}_i = \left(X^t W_i X\right)^{-1}\left(X^t W_i Y\right) \tag{2}$$

2 「R」の推定方法

Rのパッケージ「**spgwr**」に関数「**gwr**」があり、引数に最適バンド幅、座標情報の入った行列を入力すると、式 (2) を計算する。「**gwr**」の中身の一部を下に示す。

```
1    mt <- terms(formula, data = data)
2    mf <- lm(formula, data, method = "model.frame", na.action = na.fail)
3    lm <- lm(formula, data, x = TRUE, y = TRUE)
4    n <- NROW(fit.points)
5    if (is.null(colnames(fit.points)))
6        colnames(fit.points) <- c("x", "y")
7    y <- model.extract(mf, "response")
8    x <- model.matrix(mt, mf)
9    m <- NCOL(x)
10    gwr.b <- matrix(nrow = n, ncol = m)
11    gwr.se <- matrix(nrow = n, ncol = m)
12    gwr.R2 <- numeric(n)
13    gwr.e <- numeric(n)
14    yiybar <- (y - mean(y))
15    colnames(gwr.b) <- colnames(x)
16    sum.w <- numeric(n)
17    for (i in 1:n) {
18        dxs <- spDistsN1(coords, fit.points[i, ], longlat = longlat)
19        if (any(!is.finite(dxs)))
20            dxs[which(!is.finite(dxs))] <- 0
21        w.i <- gweight(dxs^2, bandwidth[i])
22        if (any(w.i < 0 | is.na(w.i)))
23            stop(paste("Invalid weights for i:", i))
24        lm.i <- lm.wfit(y = y, x = x, w = w.i)
25        sum.w[i] <- sum(w.i)
26        gwr.b[i, ] <- coefficients(lm.i)
27        ei <- residuals(lm.i)
28        gwr.e[i] <- ei[i]
29        rss <- sum(ei * w.i * ei)
30        gwr.R2[i] <- 1 - (rss/sum(yiybar * w.i * yiybar))
31        p <- lm.i$rank
32        p1 <- 1:p
33        inv.Z <- chol2inv(lm.i$qr$qr[p1, p1, drop = FALSE])
34        gwr.se[i, ] <- sqrt(diag(inv.Z) * (rss/(n - p)))
35        if (!fp.given && hatmatrix)
36            lhat[i, ] <- t(x[i, ]) %*% inv.Z %*% t(x) %*% diag(w.i)
37    }$
```

GWRモデルを推定しているのは「**lm.wfit**」(package:**stats**)であることが分かる。「**lm.wfit**」とはどのような関数なのか、中身を以下に示す。

```
1    wts <- sqrt(w)
2    z <- .Fortran("dqrls", qr = x * wts, n = n, p = p, y = y *
3        wts, ny = ny, tol = as.double(tol), coefficients = mat.or.vec(p,
4        ny), residuals = y, effects = mat.or.vec(n, ny), rank = integer(1),
5        pivot = 1:p, qraux = double(p), work = double(2 * p),
6        PACKAGE = "base")
7    coef <- z$coefficients
8    pivot <- z$pivot
9    r1 <- seq_len(z$rank)
10    dn <- colnames(x)
11    z$coefficients <- coef
12   z$residuals <- z$residuals/wts
13    z$fitted.values <- y - z$residuals
14    z$weights <- w
$
```

計算にFortranのプログラム「"dqrls"」(package:**base**)を用いていることが分かる。
"dqrls"のある場所は、Rのソースの、「/src/appl」　にある「dqrls.f」である。

```
c
c     dqrdc uses householder transformations to compute the qr
c     factorization of an n by p matrix x.  column pivoting
c     based on the 2-norms of the reduced columns may be
c     performed at the users option.
c
c     on entry
c
c        x       double precision(ldx,p), where ldx .ge. n.
c                x contains the matrix whose decomposition is to be
c                computed.
c
c        ldx     integer.
c                ldx is the leading dimension of the array x.
c
c        n       integer.
c                n is the number of rows of the matrix x.
c
c        p       integer.
c                p is the number of columns of the matrix x.
c
c        jpvt    integer(p).
c                jpvt contains integers that control the selection
c                of the pivot columns.  the k-th column x(k) of x
c                is placed in one of three classes according to the
c                value of jpvt(k).
c
c                   if jpvt(k) .gt. 0, then x(k) is an initial
c                                      column.
c
c                   if jpvt(k) .eq. 0, then x(k) is a free column.
c
c                   if jpvt(k) .lt. 0, then x(k) is a final column.
c
c                before the decomposition is computed, initial columns
c                are moved to the beginning of the array x and final
c                columns to the end.  both initial and final columns
c                are frozen in place during the computation and only
c                free columns are moved.  at the k-th stage of the
c                reduction, if x(k) is occupied by a free column
c                it is interchanged with the free column of largest
```

```
c                 reduced norm.  jpvt is not referenced if
c                 job .eq. 0.
c
c        work     double precision(p).
c                 work is a work array.  work is not referenced if
c                 job .eq. 0.
c
c        job      integer.
c                 job is an integer that initiates column pivoting.
c                 if job .eq. 0, no pivoting is done.
c                 if job .ne. 0, pivoting is done.
c
c     on return
c
c        x        x contains in its upper triangle the upper
c                 triangular matrix r of the qr factorization.
c                 below its diagonal x contains information from
c                 which the orthogonal part of the decomposition
c                 can be recovered.  note that if pivoting has
c                 been requested, the decomposition is not that
c                 of the original matrix x but that of x
c                 with its columns permuted as described by jpvt.
c
c        qraux    double precision(p).
c                 qraux contains further information required to recover
c                 the orthogonal part of the decomposition.
c
c        jpvt     jpvt(k) contains the index of the column of the
c                 original matrix that has been interchanged into
c                 the k-th column, if pivoting was requested.
c
c     linpack. this version dated 08/14/78 .
c     g.w. stewart, university of maryland, argonne national lab.
c
c     dqrdc uses the following functions and subprograms.
c
c     blas daxpy,ddot,dscal,dswap,dnrm2
c     fortran dabs,dmax1,min0,dsqrt
c
      subroutine dqrdc(x,ldx,n,p,qraux,jpvt,work,job)
      integer ldx,n,p,job
```

```
      integer jpvt(*)
      double precision x(ldx,*),qraux(*),work(*)
c
c     internal variables
c
      integer j,jp,jj, l,lp1,lup,maxj,pl,pu
      double precision maxnrm,dnrm2,tt
      double precision ddot,nrmxl,t
      logical negj,swapj
c
c
      pl = 1
      pu = 0
      if (job .eq. 0) go to 60
c
c        pivoting has been requested.  rearrange the columns
c        according to jpvt.
c
         do 20 j = 1, p
            swapj = jpvt(j) .gt. 0
            negj = jpvt(j) .lt. 0
            jpvt(j) = j
            if (negj) jpvt(j) = -j
            if (.not.swapj) go to 10
               if (j .ne. pl) call dswap(n,x(1,pl),1,x(1,j),1)
               jpvt(j) = jpvt(pl)
               jpvt(pl) = j
               pl = pl + 1
   10       continue
   20    continue
         pu = p
         do 50 jj = 1, p
            j = p - jj + 1
            if (jpvt(j) .ge. 0) go to 40
               jpvt(j) = -jpvt(j)
               if (j .eq. pu) go to 30
                  call dswap(n,x(1,pu),1,x(1,j),1)
                  jp = jpvt(pu)
                  jpvt(pu) = jpvt(j)
                  jpvt(j) = jp
   30          continue
```

```
               pu = pu - 1
   40       continue
   50    continue
   60 continue
c
c     compute the norms of the free columns.
c
      if (pu .lt. pl) go to 80
      do 70 j = pl, pu
         qraux(j) = dnrm2(n,x(1,j),1)
         work(j) = qraux(j)
   70 continue
   80 continue
c
c     perform the householder reduction of x.
c
      lup = min0(n,p)
      do 200 l = 1, lup
         if (l .lt. pl .or. l .ge. pu) go to 120
c
c           locate the column of largest norm and bring it
c           into the pivot position.
c
            maxnrm = 0.0d0
            maxj = l
            do 100 j = l, pu
               if (qraux(j) .le. maxnrm) go to 90
                  maxnrm = qraux(j)
                  maxj = j
   90          continue
  100       continue
            if (maxj .eq. l) go to 110
               call dswap(n,x(1,l),1,x(1,maxj),1)
               qraux(maxj) = qraux(l)
               work(maxj) = work(l)
               jp = jpvt(maxj)
               jpvt(maxj) = jpvt(l)
               jpvt(l) = jp
  110       continue
  120    continue
         qraux(l) = 0.0d0
```

```
         if (l .eq. n) go to 190
c
c           compute the householder transformation for column l.
c
            nrmxl = dnrm2(n-l+1,x(l,l),1)
            if (nrmxl .eq. 0.0d0) go to 180
               if (x(l,l) .ne. 0.0d0) nrmxl = dsign(nrmxl,x(l,l))
               call dscal(n-l+1,1.0d0/nrmxl,x(l,l),1)
               x(l,l) = 1.0d0 + x(l,l)
c
c              apply the transformation to the remaining columns,
c              updating the norms.
c
               lp1 = l + 1
               if (p .lt. lp1) go to 170
               do 160 j = lp1, p
                  t = -ddot(n-l+1,x(l,l),1,x(l,j),1)/x(l,l)
                  call daxpy(n-l+1,t,x(l,l),1,x(l,j),1)
                  if (j .lt. pl .or. j .gt. pu) go to 150
                  if (qraux(j) .eq. 0.0d0) go to 150
                     tt = 1.0d0 - (dabs(x(l,j))/qraux(j))**2
                     tt = dmax1(tt,0.0d0)
                     t = tt
                     tt = 1.0d0 + 0.05d0*tt*(qraux(j)/work(j))**2
                     if (tt .eq. 1.0d0) go to 130
                        qraux(j) = qraux(j)*dsqrt(t)
                     go to 140
  130                continue
                        qraux(j) = dnrm2(n-l,x(l+1,j),1)
                        work(j) = qraux(j)
  140                continue
  150             continue
  160          continue
  170          continue
c
c              save the transformation.
c
               qraux(l) = x(l,l)
               x(l,l) = -nrmxl
  180       continue
  190    continue
```

```
200 continue
    return
    end
```