Exceedingly Simple B-spline Code

Jan de Leeuw

May 5, 2015

There are many packages in `R` that provide B-spline bases (packages `splines` and `fda` come to mind). I decided to ignore all this excellent prior work and write my own, using the `.C()` interface. I know, I know, there is `Rcpp`. But ever since I first heard about `C++` in the early eighties I have convinced myself that I am too old to learn that high-level stuff. For me `R` is mainly a prototyping environment for optimization and numerics, and a convenient wrapper around compiled `C` and `FORTRAN` subroutines and libraries.

The code in this note is a straightforward `C` translation of `FORTRAN` code by Samiran Sinha, who has a nice pdf document (http://www.stat.tamu.edu/~sinha/research/note1.pdf) with some details.

The `R` function `bsplineBasis()` requires a vector `x` of values where the splines are evaluated, a spline degree `d`, and a vector of interior knots. The boundary knots are `d+1` copies of `lowknot` and `d+1` copies of `highknot`. It is assumed that the elements of `x` are between `lowknot` and `highknot`, and that the interior knots are increasing (and thus all different). There may be no interior knots, i.e. `innerknots=numeric(0)`, in which case we generate a basis for the polynomials.

We give the `R` code, the `C` code, and some simple runs.

```
bsplineBasis <-
  function (x, degree, innerknots, lowknot = min(x,innerknots), highknot = max(x,inne
rknots)) {
  innerknots <- unique (sort (innerknots))
  knots <-
  c(rep(lowknot, degree + 1), innerknots, rep(highknot, degree + 1))
  n <- length (x)
  m <- length (innerknots) + 2 * (degree + 1)
  nf <- length (innerknots) + degree + 1
  basis <- rep (0,  n * nf)
  res <- .C(
  "splinebasis", d = as.integer(degree),
  n = as.integer(n), m = as.integer (m), x = as.double (x), knots = as.double (knots)
, basis = as.double(basis)
  )
  basis <- matrix (res$basis, n, nf)
  basis <- basis[,which(colSums(basis) > 0)]
  return (basis)
  }
```

```
#include <stddef.h>
#include <stdlib.h>
```

```
#include <stdio.h>

double bs (int nknots, int nspline, int degree, double x, double * knots);
int mindex (int i, int j, int nrow);

void splinebasis (int *d, int *n, int *m, double * x, double * knots, double * basis)
{
    int mm = *m, dd = *d, nn = *n;
    int k = mm - dd - 1, i , j, ir, jr;
    for (i = 0; i < nn; i++) {
        ir = i + 1;
        if (x[i] == knots[mm - 1]) {
           basis [mindex (ir, k, nn) - 1] =  1.0;
            for (j = 0; j <  (k - 1);  j++) {
                jr = j + 1;
                basis [mindex (ir, jr, nn) - 1] = 0.0;
            }
        } else {
            for (j = 0; j < k ; j++) {
                jr = j + 1;
                basis [mindex (ir, jr, nn) - 1] = bs (mm, jr, dd + 1, x[i], knots);
            }
        }
    }
}

int mindex (int i, int j, int nrow) {
    return (j - 1) * nrow + i;
}

double bs (int nknots, int nspline, int updegree, double x, double * knots) {
    double y, y1, y2, temp1, temp2;
     if (updegree == 1) {
        if ((x >= knots[nspline - 1]) && (x < knots[nspline]))
            y = 1.0;
        else
            y = 0.0;
     }
     else {
        temp1 = 0.0;
        if ((knots[nspline + updegree - 2] - knots[nspline - 1]) > 0)
            temp1 = (x - knots[nspline - 1]) / (knots[nspline + updegree - 2] - knots
[nspline - 1]);
        temp2 = 0.0;
        if ((knots[nspline + updegree - 1] - knots[nspline]) > 0)
            temp2 = (knots[nspline + updegree - 1] - x) / (knots[nspline + updegree -
1] - knots[nspline]);
```

```
        y1 = bs(nknots, nspline, updegree - 1, x, knots);
        y2 = bs(nknots, nspline + 1, updegree - 1, x, knots);
        y =  temp1 * y1 + temp2 * y2;
    }
    return y;
}
```

```
set.seed <- 12345
x <- rnorm(10)
print(b<-bsplineBasis (x, 0, c(-2,-1,0,1,2)))
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    0    0    0    1
##  [2,]    0    0    0    1
##  [3,]    0    1    0    0
##  [4,]    0    1    0    0
##  [5,]    0    1    0    0
##  [6,]    0    0    1    0
##  [7,]    0    0    1    0
##  [8,]    0    0    1    0
##  [9,]    0    0    0    1
## [10,]    1    0    0    0
```

```
rowSums(b)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```
print(b<-bsplineBasis (x, 2, c(-2,-1,0,1,2),-3,3))
```

```
##             [,1]        [,2]      [,3]        [,4]      [,5]       [,6]
##  [1,] 0.00000000 0.000000000 0.0000000 0.000000000 0.4410854 0.55706865
##  [2,] 0.00000000 0.000000000 0.0000000 0.000000000 0.3423986 0.64272756
##  [3,] 0.00000000 0.013004617 0.6352646 0.351730830 0.0000000 0.00000000
##  [4,] 0.00000000 0.084227989 0.7419779 0.173794099 0.0000000 0.00000000
##  [5,] 0.00000000 0.002331621 0.5636247 0.434043689 0.0000000 0.00000000
##  [6,] 0.00000000 0.000000000 0.0000000 0.074742124 0.7371477 0.18811020
##  [7,] 0.00000000 0.000000000 0.0000000 0.005649418 0.5949971 0.39935343
##  [8,] 0.00000000 0.000000000 0.0000000 0.291817461 0.6803251 0.02785743
##  [9,] 0.00000000 0.000000000 0.0000000 0.000000000 0.1174929 0.74976754
## [10,] 0.00690014 0.572716869 0.4203830 0.000000000 0.0000000 0.00000000
##              [,7]
##  [1,] 0.001845918
##  [2,] 0.014873861
##  [3,] 0.000000000
##  [4,] 0.000000000
##  [5,] 0.000000000
##  [6,] 0.000000000
##  [7,] 0.000000000
##  [8,] 0.000000000
##  [9,] 0.132739526
## [10,] 0.000000000
```

```
rowSums(b)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```
print(b<-bsplineBasis (x, 4, c(-2,-1,0,1,2)))
```

```
##       [,1]         [,2]      [,3]         [,4]        [,5]        [,6]
##  [1,]    0 0.0000000000 0.0000000 0.000000e+00 0.032426060 0.282162657
##  [2,]    0 0.0000000000 0.0000000 0.000000e+00 0.019539465 0.214962181
##  [3,]    0 0.0005324626 0.2084478 5.343299e-01 0.236070762 0.020619096
##  [4,]    0 0.0223360715 0.4094825 4.563143e-01 0.106833058 0.005034065
##  [5,]    0 0.0000171163 0.1500423 5.265868e-01 0.291954887 0.031398987
##  [6,]    0 0.0000000000 0.0000000 1.207971e-03 0.149107784 0.508659540
##  [7,]    0 0.0000000000 0.0000000 6.901337e-06 0.062384684 0.382634910
##  [8,]    0 0.0000000000 0.0000000 1.841401e-02 0.328221637 0.524465003
##  [9,]    0 0.0000000000 0.0000000 0.000000e+00 0.002300765 0.056767092
## [10,]    1 0.0000000000 0.0000000 0.000000e+00 0.000000000 0.000000000
##            [,7]         [,8]         [,9]
##  [1,] 0.5271619 0.1582357249 1.362966e-05
##  [2,] 0.5301589 0.2344545663 8.849270e-04
##  [3,] 0.0000000 0.0000000000 0.000000e+00
##  [4,] 0.0000000 0.0000000000 0.000000e+00
##  [5,] 0.0000000 0.0000000000 0.000000e+00
##  [6,] 0.3233320 0.0176927228 0.000000e+00
##  [7,] 0.4752319 0.0797415821 0.000000e+00
##  [8,] 0.1285113 0.0003880183 0.000000e+00
##  [9,] 0.3524752 0.5179778105 7.047913e-02
## [10,] 0.0000000 0.0000000000 0.000000e+00
```

```
rowSums(b)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```