A very short note on B-splines

BY SAMIRAN SINHA
Department of Statistics, Texas A&M University, College Station, Texas 77843-3143
sinha@stat.tamu.edu

The present note clarifies some of the underlying facts which are used in the calculation of the basis functions of B-spline using R. Suppose we want to construct the basis functions for the cubic B-spline for a given value of \( x \), a set of inner knot points, and boundary knot points. Let the inner knot points be \( c(-0.5, 0, 0.5) \) and the boundary knot points be \( c(-4, 4) \), then the command in R to generate the spline basis functions is

```r
> library(splines)
> x=rnorm(1, 0, 1)
> x
[1] -0.2355063
> bs(x, knots=c(-0.5, 0, 0.5), Boundary.knots=c(-4, 4), degree=3, intercept=T)
```

The above command will produce 7 basis functions. Note the set of all knot points are \( c(-4, -4, -4, -4, -0.5, 0, 0.5, 4, 4, 4, 4) \) and the number of basis functions is \( m - n - 1 \), where \( m \) is the total number of knot points which is 11 in this example, and \( n \) is the degree, and for this example it is 3. Note that the knots points are ordered, and in R the entire set of knots are obtained by adding \( (n + 1) \) lower boundary knot and \( (n + 1) \) upper boundary knot with the inner knot points. Let the cubic spline basis functions be \( N_{7,3}(x) \), \( N_{6,3}(x), \cdots, N_{1,3}(x) \), where the second subscript denotes the degree of the splines. Each of the basis can be constructed through the following recursive formula. Thus to construct the basis functions of degree 3 one needs to compute all the basis functions of degree lower than 3. The recursive formula is given below.

\[
N_{i,0}(x) = I(u_i \leq x < u_{i+1})
\]

\[
N_{i,p}(x) = \frac{x - u_i}{u_{i+p} - u_i} N_{i,p-1}(x) + \frac{u_{i+p+1} - x}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(x)
\]

where \( u_i \)'s are the ordered knot points, and the degree of the spline, \( p \), will take values 1, 2, and 3. For the above computation we define \( 0/0 \) as 0. The following figure shows the necessary splines needed to compute before we get \( N_{7,3}(x) \).
Similarly to obtain the value of $N_{1,3}(x)$ one has to compute the basis which come across in the following figure.

Note that when we write

```r
> bs(x, knots=c(-0.5, 0, 0.5), Boundary.knots=c(-4, 4), degree=3, intercept=F)
```

then R will return 6 basis functions $N_{7,3}(x)$, $N_{6,3}(x)$, $\cdots$, $N_{2,3}(x)$. Following is a simple R
code to generate basis function for given inner knots and the boundary knots.

```r
newbs=function(x, degree, inner.knots, Boundary.knots) {
  Boundary.knots=sort(Boundary.knots);
  knots=c(rep(Boundary.knots[1], (degree+1)), sort(inner.knots),
          rep(Boundary.knots[2], (degree+1)));
  np=degree+length(inner.knots)+1
  s=rep(0, np)
  if(x==Boundary.knots[2]) {s[np]=1} else {for( i in 1: np)
    s[i]=basis(x, degree, i, knots)}
  return(s)
}

basis=function(x, degree, i, knots)
{ if(degree==0){ if((x<knots[i+1])&(x>=knots[i])) y=1 else
  y=0}else{
  if((knots[degree+i]-knots[i])==0) {temp1=0} else {temp1=
    (x-knots[i])/(knots[degree+i]-knots[i])};
  if((knots[i+degree+1]-knots[i+1])==0) {temp2=0} else {temp2=
    (knots[i+degree+1]-x)/(knots[i+degree+1]-knots[i+1])}
  y= temp1*basis(x, (degree-1), i, knots) +temp2*basis(x, (degree-1),
           (i+1), knots)}
  return(y)
}
> newbs(2, degree=3, inner.knots=c(-0.25, -0.5, 0, 0.25, 0.5),
   + Boundary.knots=c(-4, 4))
[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
    0.0000000 0.0000000 0.1523810 0.4252154 0.3436864 0.0787172
>
Following is the Fortran code to generate B-spline basis function.

```fortran
subroutine splinebasis(d, n, m, m1, k, x, innerknots, *boundaryknots, basis)
C This subroutine generates Bspline basis functions.
C x(n) is a n by 1 input vector for which B-spline basis
C function will be evaluated.
C innerknots(m1) set of m1 innerknot points.
C newknots is the entire set of knots, of length m=m1+2(d+1)
C where d is the degree of the splines.
C k=number of spline basis=m1+d+1
IMPLICIT NONE
integer*4 d, k, m, m1, n
double precision x(n), innerknots(m1), boundaryknots(2)
double precision newknots(m), basis(n, k), result
external b
integer*4 i1, i, j
do i1=1, (d+1)
  newknots(i1)=boundaryknots(1)
end do
do i1=(d+2), (m1+d+1)
  newknots(i1)=innerknots(i1-d-1)
end do
do i1=(m1+d+2), m
```

3
newknots(i1)=boundaryknots(2)
end do
doi=1, n
if(x(i).eq.boundaryknots(2)) then
  basis(i, k)=1.d0
  do j=1, (k-1)
    basis(i, j)=0.d0
  end do
else
  do j=1, k
    call b(m, j, (d+1), x(i), newknots, result, b)
    basis(i, j)=result
  end do
endif
end do
return
dend

C ----------------
subroutine b(i1, i2, i3, y, newknots, result, dumsub)
C This subroutine calculates i2 th basis of spline of
C degree (i3-1).
IMPLICIT NONE
integer*4 i1, i2, i3
double precision y, newknots(i1), temp1, temp2, result,
* result1, result2
external dumsub
if(i3.eq.1) then
  if((y.ge.newknots(i2)).and.(y.lt.newknots(i2+1))) then
    result=1.d0
  else
    result=0.d0
  endif
else
  call dumsub(i1, i2, (i3-1), y, newknots, result1, dumsub)
  temp1=(y-newknots(i2))*result1/(newknots(i2+i3-1)-
  * newknots(i2))
  if(temp1.ne.temp1) temp1=0.d0
  call dumsub(i1, (i2+1), (i3-1), y, newknots, result2, dumsub)
  temp2=(newknots(i2+i3)-y)*result2/
  * (newknots(i2+i3)-newknots(i2+1))
  if(temp2.ne.temp2) temp2=0.d0
  result=temp1+temp2
endif
return
dend

If one wants to call this subroutine from R following is an example code. We assume that
the above Fortran subroutines were saved in a file named “spline.f”.

> dyn.load("spline.so")
> n=10;
> m1=3;
> d=3;
> m=m1+2*(d+1);
knots = c(-0.25, 0.0, 0.25)
boundaryknots = c(-3, 3)
x = rnorm(n)
k = d + m1 + 1;
basis = matrix(0, nrow = n, ncol = k);
storage.mode(basis) <- "double"
f1 = .Fortran("splinebasis", d = as.integer(d), n = as.integer(n),
+    m = as.integer(m), m1 = as.integer(m1), +
+    k = as.integer(k), x = as.double(x), knots = as.double(knots),
+    boundaryknots = as.double(boundaryknots), output = basis)