Specifying a Natural Cubic Spline

Christian A. Maino Vieytes

Background

Using splines in regression models is a popular method for flexibly modeling exposure-outcome relationships in epidemiological studies.[1,2] Splines in a regression context can be conceptualized as a series of local polynomials fit over the domain of the regression function. It provides a nice alternative to a categorical analysis of dose response, which has several limitations. One of those key limitations is the assumption that the response is uniform over all observations within a category for which a parameter estimate is made, resulting in the appearance of a step-wise function for the relationship between exposure and response.[3] In contrast, fitting a model with spline terms is a parametric technique that generates a smooth curve, allowing the user to visualize the dose-response relationship. The parameter estimates are rarely of interest, owing to their lack of interpretability.[3] However, given that a model specifying the exposure as a continuous variable is nested in a model with an expansion of spline terms for that variable, the departure from a linear relationship can be formally tested with a Likelihood Ratio Test.[2]

Basis Representation for Cubic Splines

Fitting spline models involves applying a set of transformations to the exposure variable, X. In the regression model, we replace the original variable with the set of transformations (which we term *basis functions*) and estimate parameters for each of those transformations:

$$Y = \sum_{m=1}^{M} \beta_m h_m(X) + \gamma V$$

which is a linear basis expansion in X, where h_m is the m^{th} basis function (of which there are M), and γV is a term for an additional covariate we may want to adjust for (no basis expansion on this term).[4] We generalize this approach to the multivariable setting where we desire basis expansions in several variables:

$$Y = \sum_{j=1}^{J} \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X_j) + \gamma V$$

where j is the index of variables. The set of fixed basis functions is of interest as are the *knots* and *knot locations*, which are the locations along the domain where separate local piecewise polynomials are fit. Uniform or non-uniform spacing can be used to specify the knots, although a set of quantiles of the exposure are conventionally used as the knot locations.[5] In order to achieve smoothness in the spline curve across these disjoint regions, a continuity constraint is applied. Using a truncated power basis, we achieve the continuity constraint:

$$(x-\xi_\ell)_+^g = \begin{cases} (x-\xi_\ell)^g, & \text{if } x > \xi_\ell \\ 0, & otherwise \end{cases}$$

where ξ_ℓ is the ℓ^{th} knot and g is the order of the polynomial we choose to fit. We arrive at the number of basis functions by beginning with basis functions for the degree of the polynomial we desire and then include one truncated power basis function for for each knot we specify (an additional basis function where we apply a constant–i.e., 1–can also be included for the intercept).[5] For a cubic spline (where $g = 3$) with two interior knots, we use the following basis functions:

$$h_1(x) = 1, h_2(x) = x, h_3(x) = x^2, h_4(x) = x^3, h_5(x) = (x-\xi_1)_+^3, h_6(x) = (x-\xi_2)_+^3$$

It is shown that for a degree g polynomial, the basis functions will results in a curve that is continuous (i.e., smooth) up to the $(g-1)$ derivative.[5] For the cubic spline basis representation above, we have that Y will be continuous up to its second derivative at each of the knot boundaries. For a cubic spline, we determine the degrees of freedom by counting the number of parameters we must estimate within each interval and the number of constraints. That is, we require 4 parameters within each interval and 3 constraints (continuity in Y, Y', and Y'') at each knot. If we specify two interior knots ($K = 2$) we have:

$$4(K+1) - 3K = 4(2+1) - (3)(2) = 6 \; df$$

and $K+1 = 2+1 = 3$ intervals. We can fit the model using standard statistical software and estimation is carried out in the routine manner (e.g., least squares for linear regression or maximum likelihood estimation for generalized linear models).

Natural Cubic Splines

A cited limitation of cubic splines is the high variance in the spline estimates at the extremes of the data (i.e., in the highest and lowest intervals).[4] Natural cubic splines take cubic splines and apply two additional constraints: that the function is linear beyond the boundary knots (i.e., the exterior knots–if we have two interior knots, then we have two additional exterior or boundary knots). Y'' and Y''' will be zero at these knot boundaries and we will have K

basis functions for a natural cubic spline with K knots (K here assumes we count the two exterior or boundary knots in addition to the interior knots–for a spline with two interior knots we have $K = 4$).

In our analysis we specify the diet quality indices using a basis expansion for a natural cubic spline with two interior knots ($K = 4$). These basis functions are provided below.

In a general form[4] :

$$h_1(x) = 1, h_2(x) = x, h_{k+2}(x) = d_k(x) - d_{K-1}(x)$$
$$\text{where } d_k(x) = \frac{(x - \xi_k)_+^3 - (x - \xi_K)_+^3}{\xi_K - \xi_k} \text{ and } k \in \{1...K-2\}$$

And in the case of $K = 4$:

$$h_1(x) = 1, h_2(x) = x, h_3(x) = d_1(x) - d_3(x), h_4(x) = d_2(x) - d_3(x),$$

Implementation in R

Natural cubic splines are implemented in R using the `ns()` function from the `splines` package. The R code we specified for this analysis is found at: https://github.com/cmainov/nhanes-fi-ca-mortality/blob/main/R/utils.R. We demonstrate the use of `ns()` with a simple reproducible example. First, we generate some some data by sampling from two Gaussian distributions and then use the `ns()` function to generate a matrix with the basis functions in its columns. We want to use *two* interior knots ($K = 4$) so we will specify the `df = 3` argument. `ns()` counts the interior knots in the following way:

$$K_{interior} = df - intercept - 1$$

where $intercept \in \{0, 1\}$ and is controlled by the `intercept =` argument in the `ns()` function. Note: the default value is `intercept = FALSE`, which results in $intercept = 0$. The `lm()` function will compute the intercept for us and, thus, there is no need to redundantly specify it within `ns()`

```
# generate some toy data
set.seed( 23 ) # set seed for reproducibility
x <- rnorm( 20, mean = 35, sd = 5 )
y <- rnorm( 20, mean = 78, sd = 10 )
```

```
# specify a natural cubic spline with 2 interior knots (K=4)
# (generate the basis matrix)
k2 <- ns( x, df = 3 )

head( k2, 5 ) # print first 5 rows of the basis matrix

#  1          2          3
# [1,]  0.27013521 0.4949742 -0.31550699
# [2,] -0.07878135 0.4987137 -0.33410946
# [3,]  0.53390462 0.3255878  0.03408727
# [4,] -0.14194707 0.4300676  0.71187952
# [5,]  0.50852945 0.3222180  0.09029069
```

We then can use this matrix and specify a linear regression model, regressing `y` on the basis expansion in `x`:

```
# linear regression model with 'ns'
lm( y ~ ns( x, df = 3 ) )

# Call:
#   lm(formula = y ~ ns(x, df = 3))
#
# Coefficients:
#   (Intercept)  ns(x, df = 3)1  ns(x, df = 3)2  ns(x, df = 3)3
# 80.999           6.804         -13.666          -2.166
```

The output reflects the parameter estimates for the four basis functions we detailed above (including one for the intercept).

Note: we can supply R with the exact locations of the interior and boundary knots and get the same basis matrix (instead of specifying the `df` argument).

```
# alternative code for specifying the same matrix
k2.v2 <- ns( x,
    knots = quantile( x, c( 0.33, 0.66 ) ), # interior knots (1st and 2nd tertiles)
    Boundary.knots = quantile( x, c( 0, 1 ) ) ) # boundary knots (min and max)

head( k2.v2, 5 ) # print first 5 rows of the basis matrix

#  1          2          3
# [1,]  0.27454544 0.4970633 -0.31053853
# [2,] -0.08486486 0.5069896 -0.33380781
# [3,]  0.52668014 0.3284054  0.04392163
# [4,] -0.14508287 0.4247300  0.72035287
# [5,]  0.49944729 0.3253491  0.10026887
```

A final note: the `ns()` actually implements a slightly modified version of the basis functions for the natural cubic spline (which is why we do not get the exact values in the basis matrix that we would have otherwise predicted). Indeed, `ns()` implements a linear transformation of the basis to generate a B-spline basis matrix for the natural cubic spline we seek to fit (done for computational reasons).

References

1. Greenland, S. Dose-response and trend analysis in epidemiology: alternatives to categorical analysis. eng. *Epidemiology (Cambridge, Mass.)* **6,** 356–365. doi:10.1097/00001648-199507000-00005 (July 1995).
2. Witte, J. S. & Greenland, S. A nested approach to evaluating dose-response and trend. en. *Annals of Epidemiology* **7,** 188–193. doi:10.1016/S1047-2797(96)00159-7 (Apr. 1997).
3. Steenland, K. & Deddens, J. A. A Practical Guide to Dose-Response Analyses and Risk Assessment in Occupational Epidemiology. *Epidemiology* **15.** Publisher: Lippincott Williams & Wilkins, 63–70 (2004).
4. Hastie, T., Tibshirani, R. & Friedman, J. *The Elements of Statistical Learning* ISBN: 978-0-387-84857-0 978-0-387-84858-7. doi:10.1007/978-0-387-84858-7 (Springer New York, New York, NY, 2009).
5. James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning* ISBN: 978-1-4614-7137-0 978-1-4614-7138-7. doi:10.1007/978-1-4614-7138-7 (Springer New York, New York, NY, 2013).