General Parametric Splines in carEx

2019-06-25

Introduction

The parametric polynomial splines implemented in the 'carEx' package are piecewise polynomial functions on $k+1$ intervals formed by k knots partitioning the real line:

$$(-\infty, t_1], (t_1, t_2], ..., (t_{i-1}, t_i], ..., (t_k, \infty)$$

with degree d_i on the ith interval $(t_{i-1}, t_i]$, $i = 1, ..., k+1$, and order of continuity c_i at the ith knot, $i = 1, ..., k$.

The order of continuity refers to the highest order for which the derivatives of the polynomial on the interval to the left and to the right of a knot, t_i, have the same limits at t_i. For all orders above c_i, derivatives, if any, are not constained to have the same limit.

Such a spline is parametrized by three vectors: a vector of knots, $t_1 < t_2 < ... < t_k$, of length $k > 0$, a vector of polynomial degrees, $d_1, d_2, ..., d_{k+1}$, of length $k+1$, and a vector of orders of continuity or 'smoothness', $c_1, c_2, ..., c_k$, of length k.

Theory

We first describe the general principles that underly the implemention of splines in this package.

Let X_f be a $n \times q$ matrix for a model whose coefficients are subject to c linearly independent constraints given by a $c \times q$ matrix C. That is, the linear space for the model is:

$$\mathcal{M} = \{\eta = X_f\phi : \phi \in \mathbb{R}^q, C\phi = 0\}$$

We wish to construct a $n \times p$ design matrix X with $p = q - c$ so that

$$\mathcal{M} = \{\eta = X\beta : \beta \in \mathbb{R}^p\}$$

Supppose further that we want the parameters β to provide p specified linearly independent function of ϕ represented by the rows of the $p \times q$ matrix E whose rows are linearly independent of the rows of C to ensure that they are not equal to 0 on $\mathcal{M}$.

Consider the $q \times q$ partitioned matrix $\begin{bmatrix} C \\ E \end{bmatrix}$. Since its rows are linearly independent, it is invertible and has a conformably partitioned inverse:

$$\begin{bmatrix} F & G \end{bmatrix} = \begin{bmatrix} C \\ E \end{bmatrix}^{-1}$$

Thus $FC + GE = I$, $CF = I$, etc.

Consider the model matrix $X = X_f G$. We show that $\mathcal{M} = \{X\beta : \beta \in \mathbb{R}^p\}$ and that for any $\phi \in \mathbb{R}^q$, such that $C\phi = 0$, $\beta = E\phi$.

Suppose $C\phi = 0$. Then

$$\phi = \begin{bmatrix} F & G \end{bmatrix} \begin{bmatrix} C \\ E \end{bmatrix} \phi = \begin{bmatrix} F & G \end{bmatrix} \begin{bmatrix} 0 \\ E\phi \end{bmatrix} = GE\phi$$

Thus, with $\beta = E\phi$, we have

$$X_f\phi = X_f GE\phi = X\beta$$

We therefore have a 1-1 correspondence between $\beta \in \mathbb{R}^p$ and $\{\phi \in \mathbb{R}^q : C\phi = 0\}$ given by $\beta = E\phi$ and $\phi = G\beta$.

If X is of full rank, we can obtain the least-squares estimator $\hat{\beta} = (X'X)^{-1}X'Y$. We can then estimate any linear function $\psi = L\phi$ of ϕ under the constraint $C\phi = 0$ with the estimator $\hat{\psi} = A\hat{\beta}$ with

$$A = LG$$

Thus, the matrix G serves as a post-multiplier to transform X_f into a model matrix $X = X_f G$ that can be used in a linear model. The matrix G also serves as a post-multiplier to transform any general linear hypothesis matrix expressed in terms of ϕ into a general linear hypothesis matrix in terms of of β.

Application to Splines

Our goal is to generate model matrices for splines in a way that produces interpretable coefficients and lends itself to easily estimating and testing properties of the spline that are linear functions of parameters: slope, curvature, discontinuities, etc.

Given k knots, $-\infty = t_0 < t_1 < ... < t_k < t_{k+1} = \infty$, the spline in the ith interval, $(t_{i-1}, t_i]$, is a polynomial of degree d_i, a non-negative integer with the value 0 signifying a constant over the corresponding interval.

The order of smoothness c_i at t_i is either a non-negative integer or -1 to allow a discontinuity. (TODO: control direction of discontinuity)

Generating a model matrix for some piecewise polynomial functions is simple. For example, if the degrees, d_i, are non-decreasing and the order of continuity is a constant c less than $\min(d_i)$, one can add terms using 'plus' functions at each knot. For example, a quadratic spline (degree 2, continuity 1) with one knot at 1 can be generated with a model matrix with three columns, in addition to the intercept term:

$$x, x^2, (x-1)_+^2$$

where

$$(y)_+ = \begin{cases} 0 & \text{if } y < 0 \\ y & \text{otherwise} \end{cases}$$

A spline that is quadratic on the interval $(-\infty, 1]$ and cubic on $(1, \infty)$ with continuity of order 1, $c_1 = 1$, at $t_1 = 1$, can be generated by the columns:

$$x, x^2, (x-1)_+^2, (x-1)_+^3$$

However, if one allows the degree of the polynomial or the order of smoothness to vary in diffenret parts of the spline, the approach above works only in special cases.

Generating model matrices in more general situations, for example with degrees that are not monotone, nor monotone increasing as the index radiates from a central value, is more challenging. The approach described here works for any pattern of degrees, d_i and smoothness constraints, c_i.

We start by constucting a matrix, X_f, for a spline in which the polynomial degree in each interval is the maximal value, $\max(d_i)$. We then construct constraints for the coefficients of this model to produce the desired spline.

As an example, consider a spline, $\mathcal{S}$, with knots at 3 and 7, polynomial degrees, $(2, 3, 2)$, and smoothness, $(1, 2)$, meaning that $\mathcal{S}$ is smooth of order 1 at $x = 3$, and smooth of order 2 at $x = 7$. Columns of the full matrix X_f contain the intercept, linear and quadratic and cubic terms in each interval of the spline.

To create an instance of X_f we need to specify the values over which the matrix is evaluated. Evaluating X_f at $x = 0, 1, ...9$, we obtain the following matrix, which happens here to be block diagonal because of the ordering of the x values:

```
Xf(0:9, knots = c(3,7), degree = 3)
```

```
     X0 X1 X2 X3 X0 X1 X2  X3 X0 X1 X2  X3
f(0)  1  0  0  0  0  0  0   0  0  0  0   0
f(1)  1  1  1  1  0  0  0   0  0  0  0   0
f(2)  1  2  4  8  0  0  0   0  0  0  0   0
f(3)  1  3  9 27  0  0  0   0  0  0  0   0
f(4)  0  0  0  0  1  4 16  64  0  0  0   0
f(5)  0  0  0  0  1  5 25 125  0  0  0   0
f(6)  0  0  0  0  1  6 36 216  0  0  0   0
f(7)  0  0  0  0  1  7 49 343  0  0  0   0
f(8)  0  0  0  0  0  0  0   0  1  8 64 512
f(9)  0  0  0  0  0  0  0   0  1  9 81 729
```

The model for the unconstrained maximal polynomial is $X_f\phi : \phi \in \mathbb{R}^{12}$.

We impose three types of constraints on ϕ.

1. $X_f\phi$ should evaluate to 0 at $x = 0$ so an intercept term in the model will have the correct interpretation,
2. the limits of the value and of the first derivative of the spline must be the same when approaching the first knot from the right or from the left, and the limits of the value, the first and second derivatives should be the same when approaching the second knot from the right or from the left, and
3. the degree of the polynomial in the first and third intervals must be reduced to 2.

The constraint marix, C is created by the 'Cmat' function:

```
Cmat(knots = c(3, 7), degree = c(2, 3, 2), smooth = c(1, 2))
```

```
       X0 X1 X2  X3 X0 X1  X2   X3 X0 X1 X2  X3
f(0)    1  0  0   0  0  0   0    0  0  0  0   0
C(3).0 -1 -3 -9 -27  1  3   9   27  0  0  0   0
C(3).1  0 -1 -6 -27  0  1   6   27  0  0  0   0
C(7).0  0  0  0   0 -1 -7 -49 -343  1  7 49 343
C(7).1  0  0  0   0  0 -1 -14 -147  0  1 14 147
C(7).2  0  0  0   0  0  0  -2  -42  0  0  2  42
I.1.3   0  0  0   1  0  0   0    0  0  0  0   0
I.3.3   0  0  0   0  0  0   0    0  0  0  0   1
attr(,"ranks")
  npar.full         C.n      C.rank spline.rank
         12           8           8           4
attr(,"d")
[1] 536.66701452  48.80391245  10.85308819   3.18591258   0.97504352
[6]   0.81688866   0.35905212   0.08458296
```

The row labels of the constraint matrix show the role of each row. For example, "f(0)" is the value of the spline when $x = 0$ which is constrained to 0 so that an intercept term in a linear model can have its usual interpretation, "C(3).0" ensures continuity at $x = 3$, "C(7).2" forces continuity of the second derivative at $x = 7$, "I.1.3" constrains the cubic term to be 0 in the first interval, etc.

Attributes give the length of the ϕ vector as 'npar.full', the number of constraints as 'C.n', the rank of the constraint matrix as 'C.rank' and the rank of the spline, omitting the intercept term, as 'spline.rank'.

The 'd' attribute contains the vector of singular values of the constraint matrix.

The following is the matrix E of estimable functions created by the 'Emat' function:

```
Emat(knots = c(3, 7), degree = c(2, 3, 2), smooth = c(1, 2))
```

```
     X0 X1 X2  X3 X0 X1 X2 X3 X0 X1 X2 X3
```

```
D1|0  0  1  0   0  0  0  0  0  0  0  0  0
D2|0  0  0  2   0  0  0  0  0  0  0  0  0
C2|3  0  0 -2 -18  0  0  2 18  0  0  0  0
C3|3  0  0  0  -6  0  0  0  6  0  0  0  0
```

The row labels signify the first derivative at $x = 0$, 'D1(0)', the second derivative at $x = 0$, 'D2(0)', the saltus in the second derivative at $x = 3$ and the saltus in the third derivative at $x = 3$.

The full rank model for the spline is generated by a matrix $X = X_f G$ as described in the previous section.

The spline modelling function is a closure generated by the `gspline` function.

```
sp <- gspline(knots = c(3, 7), degree = c(2, 3, 2), smoothness = c(1, 2))
sp(0:9)
```

```
     D1|0 D2|0 C2|3     C3|3
f(0)    0  0.0  0.0  0.00000
f(1)    1  0.5  0.0  0.00000
f(2)    2  2.0  0.0  0.00000
f(3)    3  4.5  0.0  0.00000
f(4)    4  8.0  0.5  0.16667
f(5)    5 12.5  2.0  1.33333
f(6)    6 18.0  4.5  4.50000
f(7)    7 24.5  8.0 10.66667
f(8)    8 32.0 12.5 20.66667
f(9)    9 40.5 18.0 34.66667
```

produce a matrix $X = X_f G$ that will generate the desired spline parametrized by linear estimable coefficients.

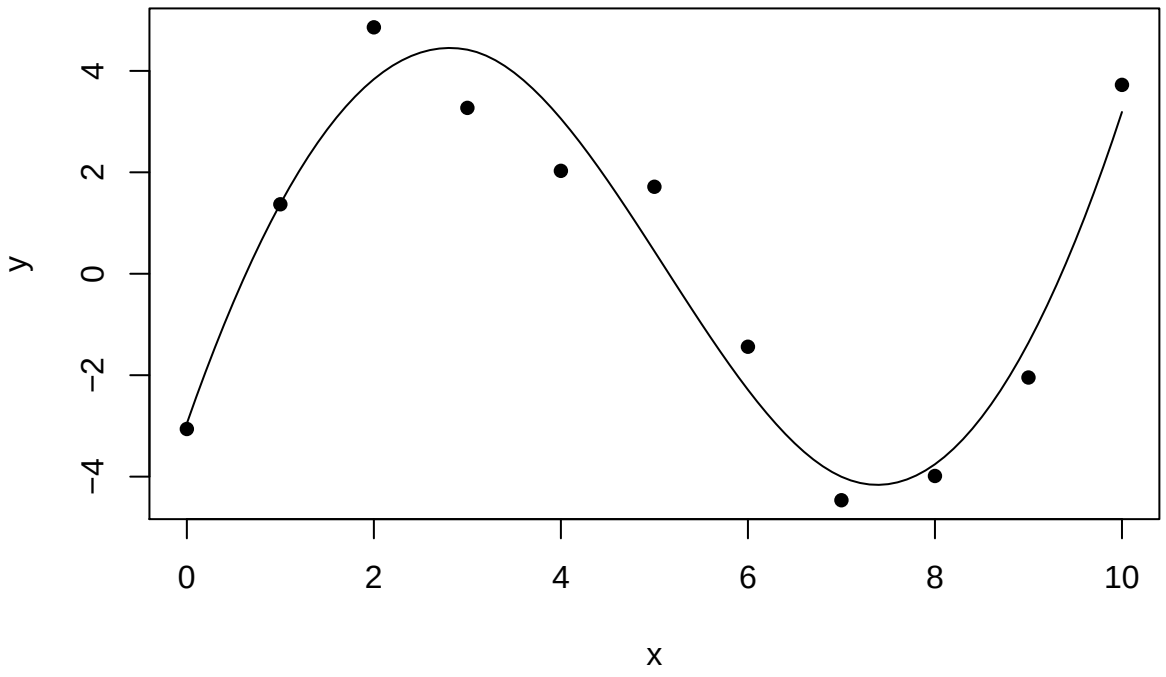
The closure created by the `gspline` function can be used in a linear model formulas. We illustrate its use with a small example. Note that the spline function can be used in any linear model formula. It can, for example, be modelled as interacting with other predictors.

```
df <- data.frame(x = 0:10)
set.seed(123)
df <- within(df, y <- -2* (x-5) + .1 * (x-5)^3 + rnorm(x))
df <- rbind(df, data.frame(x = seq(0,10,.1), y = NA))
df <- df[order(df$x),]
plot(y~x, df, pch = 16)
fit <- lm(y ~ sp(x), data = df)
summary(fit)
```

```

Call:
lm(formula = y ~ sp(x), data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-1.1476 -0.5748 -0.1091  0.6914  1.2704

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.9513     1.0165  -2.903  0.02721 *
sp(x)D1|0     5.2685     1.3117   4.017  0.00699 **
sp(x)D2|0    -1.8747     0.6726  -2.787  0.03169 *
sp(x)C2|3    -0.5129     1.3846  -0.370  0.72381
sp(x)C3|3     1.1346     0.2749   4.127  0.00616 **
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.064 on 6 degrees of freedom
  (101 observations deleted due to missingness)
Multiple R-squared:  0.9372,    Adjusted R-squared:  0.8954
F-statistic:  22.4 on 4 and 6 DF,  p-value: 0.0009419
```

```
lines(df$x , predict(fit, df))
```



Linear hypotheses

Linear hypotheses about a spline may be easy to formulate in terms of its 'full' parameter vector ϕ but challenging in terms of the 'working' parameters, β. For example, the derivative or curvature of the spline over a range of values is easily expressed in terms of ϕ. To do this We use the relationship between linear hypotheses in terms of ϕ with those in terms of β to generate linear hypotheses based on $\hat{\beta}$. Namely the least-squares estimator of $\psi = L\phi$ under the contraint $C\phi = 0$ is $\hat{\psi} = A\hat{\beta}$ where $A = LG$.

Given a spline function `sp` created by the `gspline` function:

```
sp <- gspline(knots = c(3,7), degree = c(2,3,2), smoothness = c(1,2))
sp(0:9)
```

```
     D1|0 D2|0 C2|3     C3|3
f(0)    0  0.0  0.0  0.00000
f(1)    1  0.5  0.0  0.00000
f(2)    2  2.0  0.0  0.00000
f(3)    3  4.5  0.0  0.00000
f(4)    4  8.0  0.5  0.16667
f(5)    5 12.5  2.0  1.33333
f(6)    6 18.0  4.5  4.50000
f(7)    7 24.5  8.0 10.66667
f(8)    8 32.0 12.5 20.66667
f(9)    9 40.5 18.0 34.66667
```

The `sp` function will generate a hypothesis matrix to query values and derivatives of the spline.

```
sp(c(2, 3, 7), D = 1)
```

```
     D1|0 D2|0 C2|3 C3|3
D1|2    1    2    0    0
D1|3    1    3    0    0
D1|7    1    7    4    8
```

Denoting the matrix above by A, $A\hat{\beta}$ will estimate the first derivative of the spline at $x = 2$ and its limit from the right at the knots $x = 3, 7$. The `limit` parameter to the spline function is used to select whether the value estimated is a limit from the right, from the left, or the saltus (jump) in value if discontinuous. For example, at $x = 3$ where the spline has a discontinuous second derivatives:

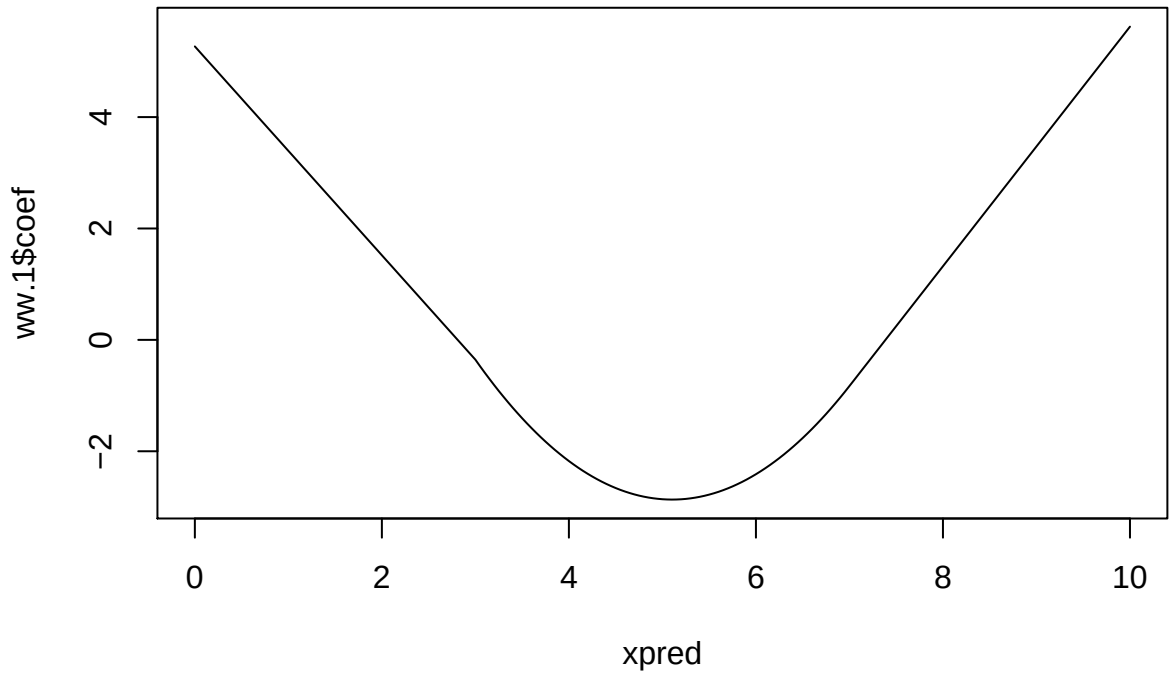
```
sp(c(3, 3, 3), D = 2, limit = c(-1,0,1))
```

```
            D1|0 D2|0 C2|3 C3|3
D2|3-          0    1    0    0
D2|3+-D2|3-    0    0    1    0
D2|3+          0    1    1    0
```

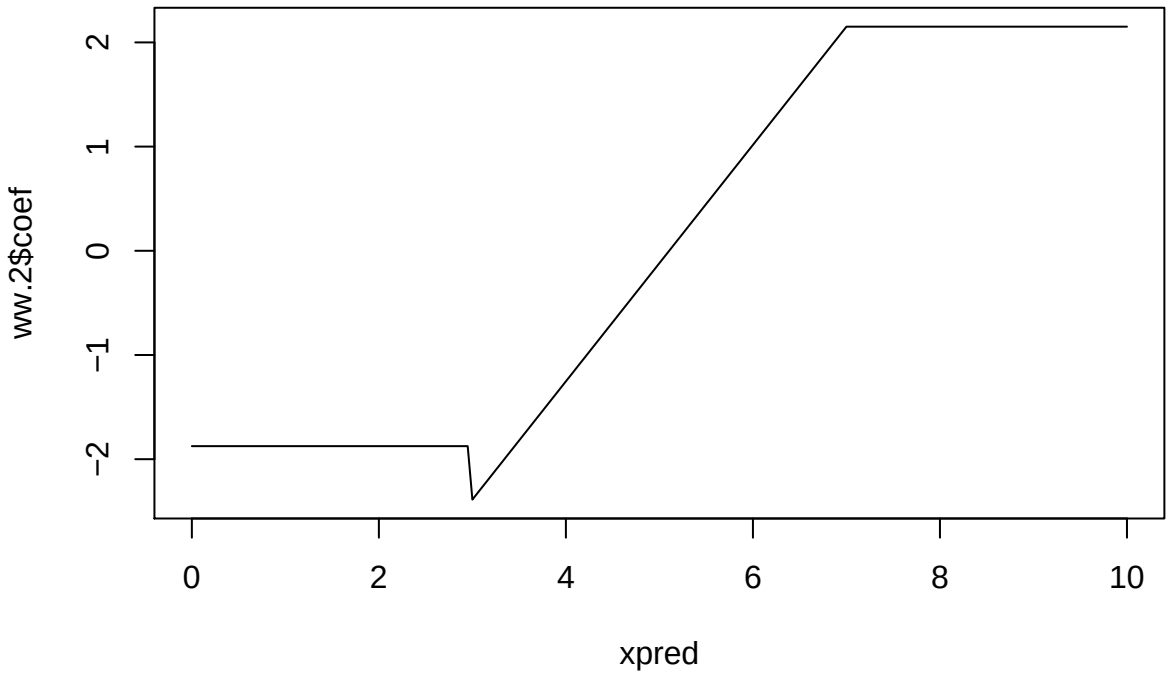
Using the 'wald' function it is possible to graph these estimates as a function of of x.

```
# customized panel function
gpanel.fit <- function (x, y, fit, lower, upper, subscripts, ..., type, group.number,
    alpha, col, col.line, col.symbol, border = F, font, fontface) {
    if (!missing(fit)) {
        if (missing(col))
            col <- "blue"
        if (!missing(group.number)) {
            col <- col.line
        }
        if (!missing(subscripts)) {
            fit <- fit[subscripts]
        }
        dd <- data.frame(x = x, fit = fit)
        dd <- dd[order(dd$x), ]
        panel.lines(dd$x, dd$fit, ..., col = col, type = "l")
    }
    if (!missing(lower)) {
        if (missing(alpha) || alpha == 1)
            alpha <- 0.3
        if (missing(col))
            col <- "blue"
        if (!missing(group.number)) {
            col <- col.symbol
        }
        if (!missing(subscripts)) {
            upper <- upper[subscripts]
            lower <- lower[subscripts]
        }
        dd <- data.frame(x = x, lower = lower, upper = upper)
        dd <- dd[order(dd$x), ]
        panel.polygon(c(dd$x, rev(dd$x)), c(dd$upper, rev(dd$lower)),
            border = border, col = col, alpha = alpha, ...)
    }
}
```

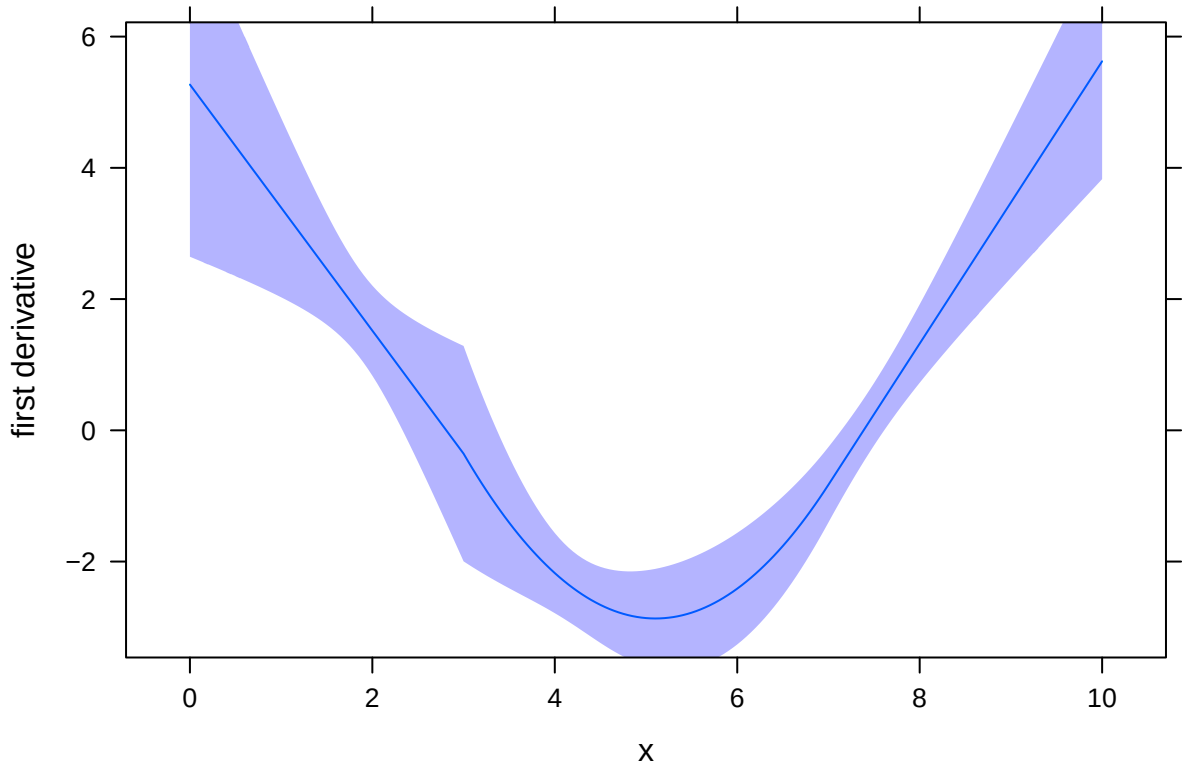
```r
xpred <- seq(0,10, .05)

A.1 <- cbind(0, sp(xpred, D = 1))
ww.1 <- as.data.frame(wald(fit, A.1))

A.2 <- cbind(0, sp(xpred, D = 2))
ww.2 <- as.data.frame(wald(fit, A.2))

plot(xpred, ww.1$coef, type = 'l')
```




```r
plot(xpred, ww.2$coef, type = 'l')
```




```
library(latticeExtra)
ww.1$x <- xpred
xyplot(coef ~ x, ww.1, type = 'l',
       lower = ww.1$L2, upper = ww.1$U2,
       ylab = 'first derivative',
       subscripts = TRUE) +
    layer(gpanel.fit(...))
```



```
head(ww.1)
```

```
            coef       se       U2       L2    x
D1|0    5.268497 1.311704 7.891905 2.645089 0.00
D1|0.05 5.174763 1.279216 7.733194 2.616331 0.05
D1|0.1  5.081028 1.246788 7.574604 2.587453 0.10
D1|0.15 4.987294 1.214425 7.416144 2.558444 0.15
D1|0.2  4.893560 1.182133 7.257826 2.529293 0.20
D1|0.25 4.799825 1.149918 7.099661 2.499989 0.25
```

Periodic splines

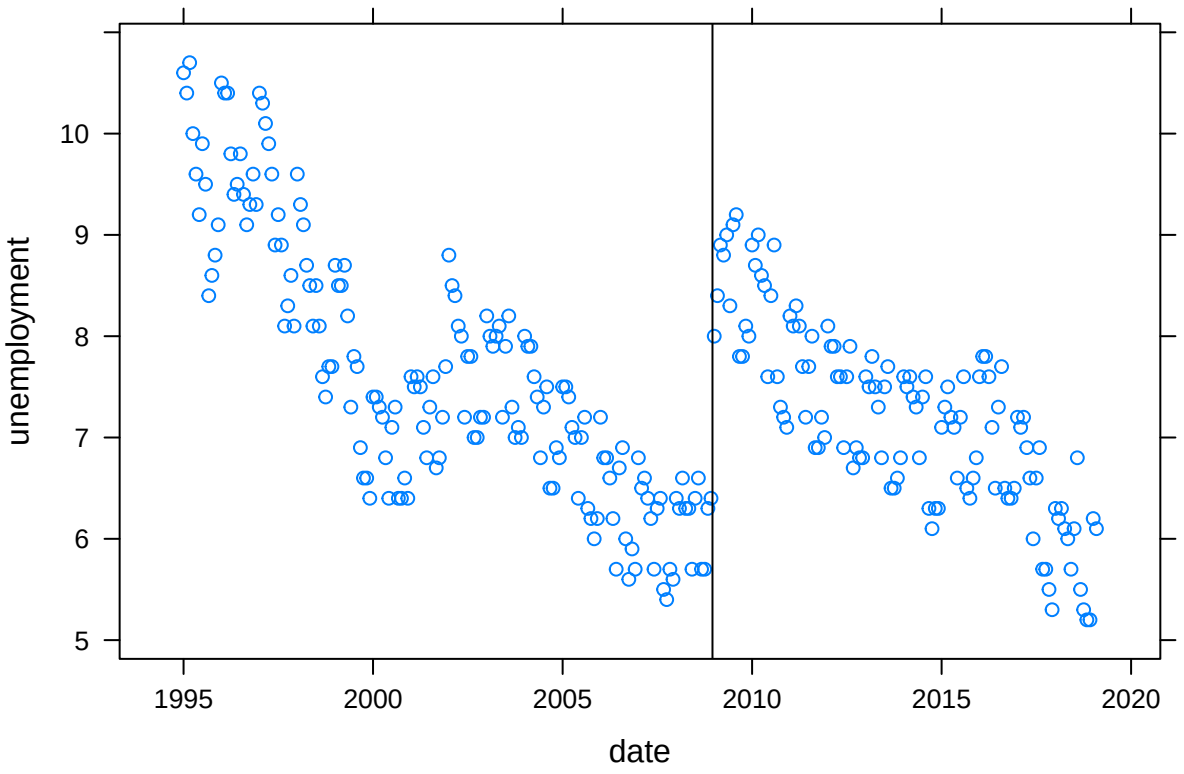
We show how periodic splines can be used to fit periodic patterns such as seasonal patterns. We use the monthly U.S. unemployment rates from January 1995 to February 2019.

The 'crash' in November 2008 creates a discontinuity in the series which we can use to illustrate how to a discontinuity at an a priori value of time. We model the series as consisting of two components, a long-term secular component and a periodic annual component. We also illustrate how to use a parsimonious secular spline to model secular interactions with the seasonal spline.

```
unemp <- Unemployment
head(unemp)
```

```
        date unemployment
1 1995-01-01         10.6
2 1995-02-01         10.4
3 1995-03-01         10.7
4 1995-04-01         10.0
5 1995-05-01          9.6
6 1995-06-01          9.2
```

```
library(lattice)
library(latticeExtra)
xyplot(unemployment ~ date, unemp) + layer(panel.abline(v = as.Date('2008-12-15')))
```




```
toyear <- function(x) {
  # number of years from January 1, 2000
  (as.numeric(x) - as.numeric(as.Date('2000-01-01')))/365.25
}
unemp <- within(
  unemp,
  {
    year <- toyear(date)
    month <- as.numeric(format(date, '%m'))
  })
summary(unemp)
```

```
      date             unemployment        month             year
 Min.   :1995-01-01   Min.   : 5.200   Min.   : 1.000   Min.   :-4.999
 1st Qu.:2001-01-08   1st Qu.: 6.600   1st Qu.: 3.000   1st Qu.: 1.023
 Median :2007-01-16   Median : 7.300   Median : 6.000   Median : 7.043
 Mean   :2007-01-15   Mean   : 7.448   Mean   : 6.466   Mean   : 7.041
 3rd Qu.:2013-01-24   3rd Qu.: 8.100   3rd Qu.: 9.000   3rd Qu.:13.066
 Max.   :2019-02-01   Max.   :10.700   Max.   :12.000   Max.   :19.086
```

```
quintiles <- quantile(unemp$year, 1:4/5)
sp2 <- gspline(quintiles, 2, 1) # quadratic spline
sp3 <- gspline(quintiles, 3, 2) # cubic spline

quintiles_with_crash <- sort(c(quintiles, toyear(as.Date('2008-12-15'))))

sp2d <- gspline(quintiles_with_crash, 2, c(1,1,-1,1,1))
sp3d <- gspline(quintiles_with_crash, 3,  c(2,2,-1,2,2))

fit2 <- lm(unemployment ~ sp2(year), unemp)
```

```
summary(fit2)
```

```
Call:
lm(formula = unemployment ~ sp2(year), data = unemp)

Residuals:
     Min       1Q   Median       3Q      Max
-1.51604 -0.49821 -0.01906  0.47250  1.90185

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        7.82474    0.09008  86.868  < 2e-16 ***
sp2(year)D1|0     -0.17468    0.06700  -2.607  0.00961 **
sp2(year)D2|0     -0.01409    0.02529  -0.557  0.57785
sp2(year)C2|-0.184 -0.17031    0.06748  -2.524  0.01216 *
sp2(year)C2|4.63   0.14677    0.04567   3.214  0.00146 **
sp2(year)C2|9.45  -0.28896    0.04569  -6.324  9.9e-10 ***
sp2(year)C2|14.3   0.17277    0.06750   2.560  0.01100 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7104 on 283 degrees of freedom
Multiple R-squared:  0.6344,    Adjusted R-squared:  0.6266
F-statistic: 81.84 on 6 and 283 DF,  p-value: < 2.2e-16
```

```
unemp$fit2 <- predict(fit2)
fit3 <- lm(unemployment ~ sp3(year), unemp)
summary(fit3)
```

```
Call:
lm(formula = unemployment ~ sp3(year), data = unemp)

Residuals:
     Min       1Q   Median       3Q      Max
-1.65225 -0.53935  0.01539  0.51200  1.94448

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         7.728274   0.115711  66.789  < 2e-16 ***
sp3(year)D1|0      -0.376882   0.047831  -7.879 7.19e-14 ***
sp3(year)D2|0       0.104451   0.055240   1.891   0.0597 .
sp3(year)D3|0      -0.011672   0.019865  -0.588   0.5573
sp3(year)C3|-0.184 -0.057021   0.071984  -0.792   0.4289
sp3(year)C3|4.63    0.002714   0.034758   0.078   0.9378
sp3(year)C3|9.45   -0.018936   0.034765  -0.545   0.5864
sp3(year)C3|14.3    0.058433   0.071957   0.812   0.4174
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7385 on 282 degrees of freedom
Multiple R-squared:  0.6062,    Adjusted R-squared:  0.5965
F-statistic: 62.02 on 7 and 282 DF,  p-value: < 2.2e-16
```

```r
unemp$fit3 <- predict(fit3)

fit2d <- lm(unemployment ~ sp2d(year), unemp)
summary(fit2d)
```

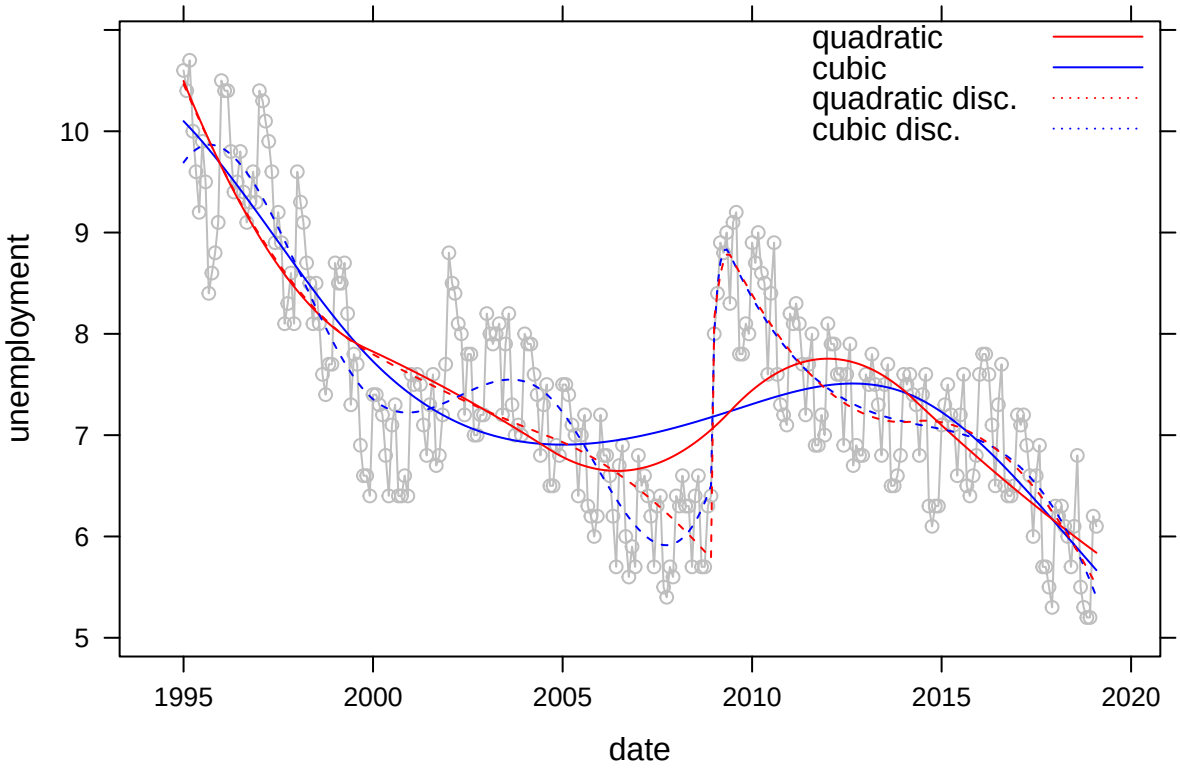
```
Call:
lm(formula = unemployment ~ sp2d(year), data = unemp)

Residuals:
     Min       1Q   Median       3Q      Max
-1.50968 -0.44994  0.05091  0.45359  1.41990

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          7.79826    0.07585 102.815  < 2e-16 ***
sp2d(year)D1|0      -0.20751    0.05896  -3.520 0.000504 ***
sp2d(year)D2|0       0.01405    0.02526   0.556 0.578491
sp2d(year)C2|-0.184 -0.12543    0.06087  -2.061 0.040267 *
sp2d(year)C2|4.63   -0.07886    0.07068  -1.116 0.265540
sp2d(year)C0|8.96    2.21150    0.61367   3.604 0.000371 ***
sp2d(year)C1|8.96    4.32041    2.67733   1.614 0.107719
sp2d(year)C2|8.96   -9.30985    5.55410  -1.676 0.094813 .
sp2d(year)C2|9.45    9.53430    5.57228   1.711 0.088184 .
sp2d(year)C2|14.3   -0.31857    0.07899  -4.033 7.11e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5864 on 280 degrees of freedom
Multiple R-squared:  0.7535,    Adjusted R-squared:  0.7455
F-statistic: 95.08 on 9 and 280 DF,  p-value: < 2.2e-16
```

```r
unemp$fit2d <- predict(fit2d)
fit3d <- lm(unemployment ~ sp3d(year), unemp)
summary(fit3d)
```

```
Call:
lm(formula = unemployment ~ sp3d(year), data = unemp)

Residuals:
     Min       1Q   Median       3Q      Max
-1.46661 -0.43416  0.04281  0.39201  1.46187

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          7.35489    0.08890  82.736  < 2e-16 ***
sp3d(year)D1|0      -0.31782    0.03501  -9.079  < 2e-16 ***
sp3d(year)D2|0       0.43791    0.04935   8.874  < 2e-16 ***
sp3d(year)D3|0      -0.19288    0.02219  -8.692 3.13e-16 ***
sp3d(year)C3|-0.184 -0.46972    0.06367  -7.377 1.87e-12 ***
sp3d(year)C3|4.63    0.56217    0.07847   7.164 7.03e-12 ***
sp3d(year)C0|8.96    1.12727    0.68856   1.637   0.1027
sp3d(year)C1|8.96    7.17553    4.63633   1.548   0.1228
```

```
sp3d(year)C2|8.96   -38.30263   19.38613  -1.976   0.0492 *
sp3d(year)C3|8.96    75.58355   39.50921   1.913   0.0568 .
sp3d(year)C3|9.45   -76.00194   39.53243  -1.923   0.0556 .
sp3d(year)C3|14.3    -0.04238    0.10074  -0.421   0.6743
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5308 on 278 degrees of freedom
Multiple R-squared:  0.7995,    Adjusted R-squared:  0.7916
F-statistic: 100.8 on 11 and 278 DF,  p-value: < 2.2e-16
```

```
unemp$fit3d <- predict(fit3d)

pp <- xyplot(unemployment ~ date, unemp, type = 'b',
             col = 'gray',
             key = list(
               corner = c(1,1),
               text = list(lab = c('quadratic','cubic','quadratic disc.','cubic disc.')),
               lines = list(col= c('red','blue','red','blue'),
                            lty = c(1,1,3,3))
               )) +
  layer(panel.lines(x, unemp$fit3, col = 'blue')) +
  layer(panel.lines(x, unemp$fit2, col = 'red')) +
  layer(panel.lines(x, unemp$fit3d, col = 'blue', lty = 2)) +
  layer(panel.lines(x, unemp$fit2d, col = 'red', lty = 2))
pp
```



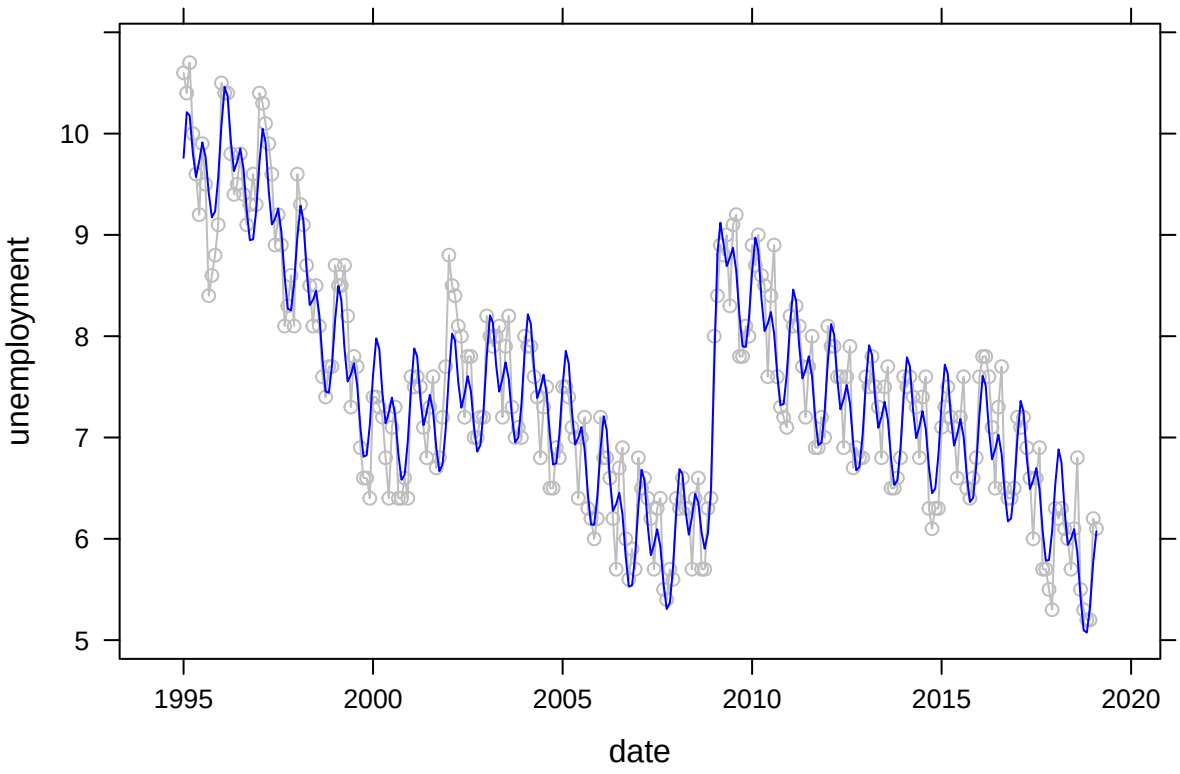
Periodic spline

We add a periodic spline component as a function of months using a cubic spline with period 12 and four internal knot at $12 \times (1/52/53/54/5)$.

```
per3 <- gspline(12 * 1:5/5, 3, 2, periodic = TRUE)
fitper3 <-  lm(unemployment ~ sp3d(year) + per3(month), unemp)
summary(fitper3)
```

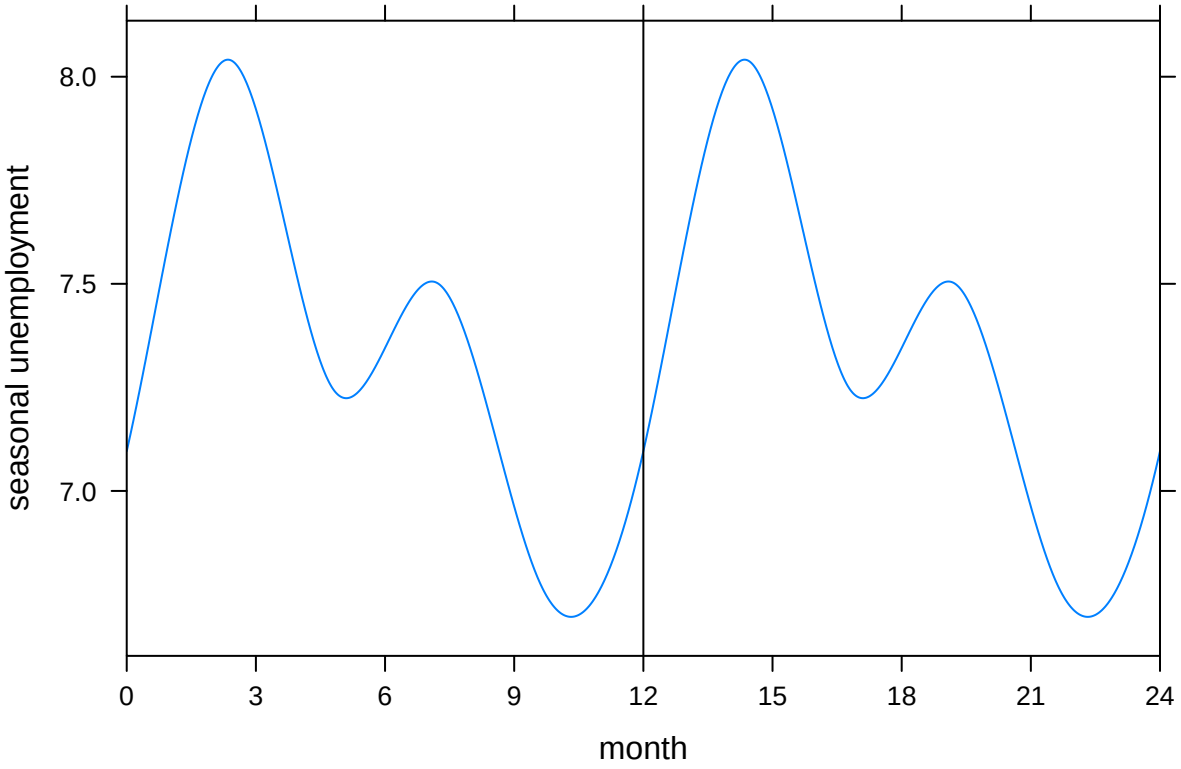
```

Call:
lm(formula = unemployment ~ sp3d(year) + per3(month), data = unemp)

Residuals:
     Min       1Q   Median       3Q      Max
-1.01225 -0.23279 -0.00892  0.20521  1.18149

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)             7.09603    0.07537  94.143  < 2e-16 ***
sp3d(year)D1|0         -0.32959    0.02339 -14.091  < 2e-16 ***
sp3d(year)D2|0          0.46948    0.03300  14.225  < 2e-16 ***
sp3d(year)D3|0         -0.20769    0.01484 -13.994  < 2e-16 ***
sp3d(year)C3|-0.184    -0.52094    0.04262 -12.223  < 2e-16 ***
sp3d(year)C3|4.63       0.62740    0.05253  11.942  < 2e-16 ***
sp3d(year)C0|8.96       0.11441    0.46534   0.246 0.805973
sp3d(year)C1|8.96      11.37552    3.12632   3.639 0.000327 ***
sp3d(year)C2|8.96     -56.50467   13.07145  -4.323 2.16e-05 ***
sp3d(year)C3|8.96     112.04499   26.63734   4.206 3.52e-05 ***
sp3d(year)C3|9.45    -112.50572   26.65288  -4.221 3.31e-05 ***
sp3d(year)C3|14.3      -0.05644    0.06728  -0.839 0.402317
per3(month)D1|12/12     0.45981    0.02659  17.290  < 2e-16 ***
per3(month)D2|12/12     0.24369    0.05616   4.339 2.02e-05 ***
per3(month)D3|12-/12   -0.03609    0.04414  -0.818 0.414265
per3(month)C3|2.4/2.4   0.87800    0.08373  10.486  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3545 on 274 degrees of freedom
Multiple R-squared:  0.9119,    Adjusted R-squared:  0.907
F-statistic:   189 on 15 and 274 DF,  p-value: < 2.2e-16
```

```
unemp$fitper3 <- predict(fitper3)
pp <- xyplot(unemployment ~ date, unemp, type = 'b',
             col = 'gray') +
  layer(panel.lines(x, unemp$fitper3, col = 'blue'))
pp
```



We can examine the monthly periodic spline fit

```
pred <- data.frame(month = seq(0,24,.01), year = 0)
pred$fitper3 <- predict(fitper3, newdata = pred)
xyplot(fitper3 ~ month, pred, type = 'l',
       xlim = c(0,24),
       scales = list( x = list( at =3 * 0:8)),
       ylab = 'seasonal unemployment') +
  layer(panel.abline(v = 12))
```


seasonal unemployment
8.0
7.5
7.0
0
3
6
9
12
15
18
21
24
month