

# Tools for General Parametric Splines

2  
'spida' package in R developed for  
the Summer Program In Data Analysis (SPIDA)  
at York University

Contact: Georges Monette ([georges@yorku.ca](mailto:georges@yorku.ca))

## Flexible tools to explore general parametric splines:

- Splines are conceptually simple but generally awkward to use in statistical analyses
- Goal: make splines as easy to use in regression as categorical variables
- “General parametric spline:”
  - Polynomials of different degrees in each interval
  - Different degrees of smoothness at each knot – including possible discontinuity
- Natural parametrization with interpretable coefficients:
  - can use regular expression to test groups of coefficients
- Can estimate and test features of spline: slope, curvature, salti.
- Can interact with other variables (numerical or categorical), other splines, etc.
- Limitation: need to centre and scale variable

# Installation

- 1) Install R <http://cran.r-project.org/>

Only once: Install 'car' and 'spida'. After starting R:

```
>1 install.packages('car')  
> install.packages('spida',  
  repos = 'http://r-forge.r-project.org')
```

- 2) Each time you run R: load spida  
> `library(spida)`

---

<sup>1</sup> The '>' sign is the prompt in R. Type only what comes after the '>'.

# Main Functions:

- gsp** Creates general parametric splines
- wald** Uses regular expressions or hypothesis matrices to estimate coefficients and test hypotheses
- Lmat** Generates hypothesis matrices
- sc** Operates on spline functions to generate hypothesis matrices for spline derivatives: slope, curvature, salti at knots
- smsp** Non-parametric splines using mixed models

## Example:

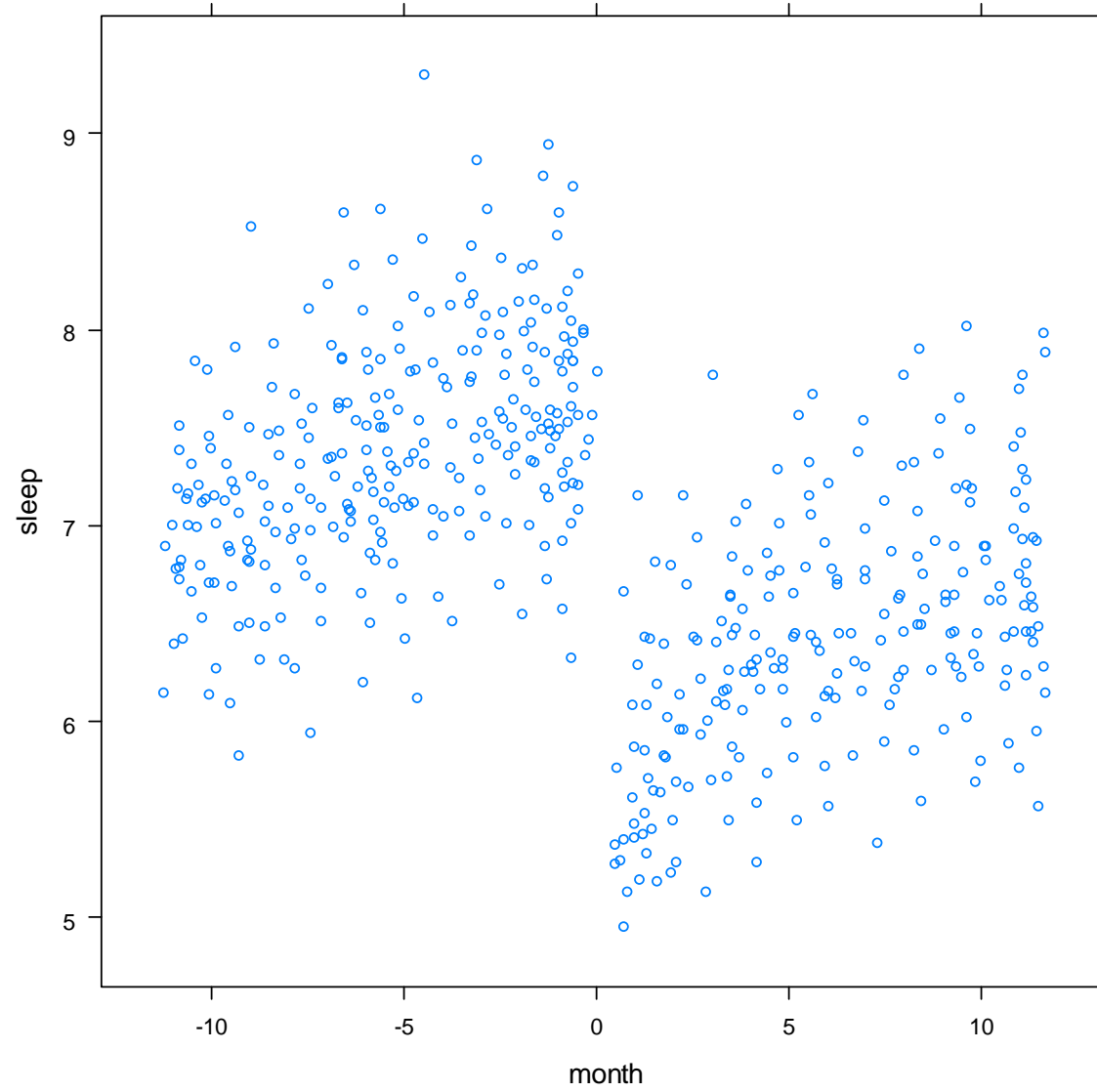
Simulated sleep data before and after birth.

Inspired by Kunkel, Reitav, Monette (2010)

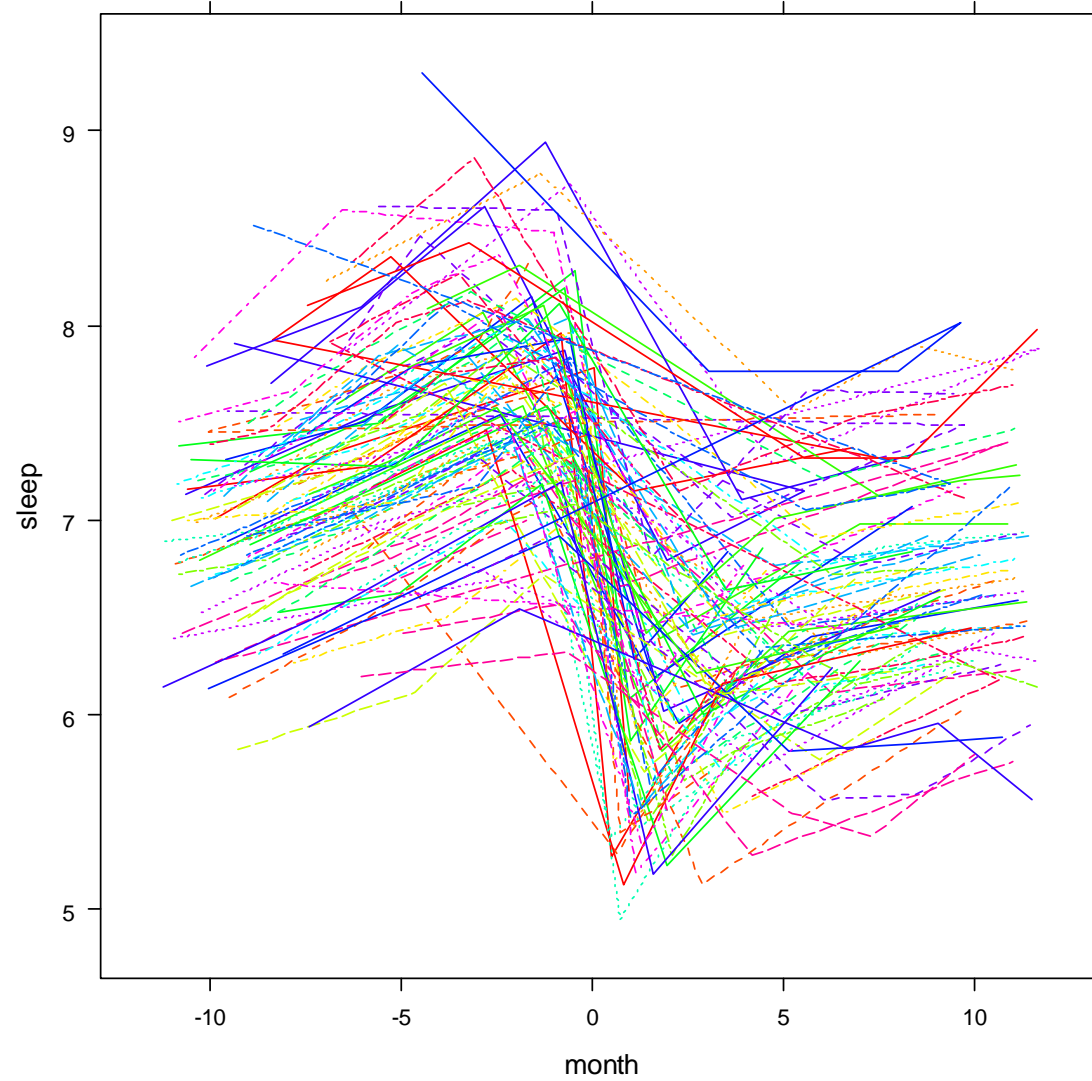
100 women each observed 5 times:

```
> sim.data <- read.csv(
  'http://www.math.yorku.ca/~georges/Data/sim.data.csv')
> dim(sim.data)
[1] 500  4
> head( sim.data)
  X      month id  sleep
1 1 -9.86370968  1 7.008448
2 2 -6.95388563  1 7.336828
3 3 -2.11407625  1 7.642227
4 4  0.05344575  1 7.778133
5 5  0.97390029  1 5.609873
6 6 -7.66649560  2 7.184266
```

```
> xyplot( sleep ~ month, sim.data)
```



```
> td( col = rainbow(20))  
> xyplot( sleep ~ month, sim.data,  
          groups = id, type = 'l')
```







# Defining a spline function:


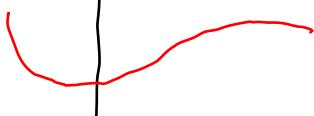

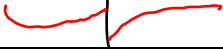
*Create your own spline function*

```
> sp <- function( x ) gsp( x,  
  knots = c(-5, 0, 5), # position of knots  
  degree = c(2, 2, 2, 2), # quad in each interval  
  smooth = c(1,-1,1)) # smoothness at knot
```

**degree** parameter:

1,2,3,...	linear, quadratic, cubic, etc.
0	flat
-1	equal to 0

**smooth** parameter:

1	'smooth': same slope on both sides of knot	
2	same curvature on both sides	
0	continuous (no gap)	
-1	discontinuous	


## What `sp` does:

Generates a portion of a model matrix:


```
> sp( c(-5,0,5) )
```

	D1(0)	D2(0)	C(0).0	C(0).1	C(0).2	C(5).2	C(-5).2
f(-5)	-5	12.5	0	0	0.0	0	0
f(0)	0	0.0	0	0	0.0	0	0
f(5)	5	12.5	1	5	12.5	0	0


linear  
term



saltus  
in level  
at 0



change  
in quad.  
term at 5



Using your **sp** in a linear multilevel mixed model:

```
> library( nlme )  
> fit <- lme( sleep ~ sp(month), sim.data,  
             random = ~ 1 | id )
```

```
> summary( fit )
```

Linear mixed-effects model fit by REML

Data: sim.data

	AIC	BIC	logLik
	384.8794	426.8642	-182.4397

Random effects:

Formula: ~1 | id

	(Intercept)	Residual
--	-------------	----------

StdDev:	0.4922676	0.2439248
---------	-----------	-----------

. . . .

**Fixed effects: sleep ~ sp(month)**

	Value	Std.Error	DF	t-value	p-value
(Intercept)	7.770252	0.07562772	393	102.74344	0.0000
sp(month)D1(0)	0.007316	0.04231428	393	0.17289	0.8628
sp(month)D2(0)	-0.027195	0.01204080	393	-2.25853	0.0245
sp(month)C(0).0	-2.284334	0.10143225	393	-22.52078	0.0000
sp(month)C(0).1	0.295341	0.06867966	393	4.30027	0.0000
sp(month)C(0).2	-0.015195	0.01850084	393	-0.82130	0.4120
sp(month)C(5).2	0.027771	0.02041699	393	1.36018	0.1746
sp(month)C(-5).2	-0.048239	0.01943028	393	-2.48267	0.0135

**Standardized Within-Group Residuals:**

Min	Q1	Med	Q3	Max
-6.955982485	-0.326223723	-0.009829296	0.320549996	7.518591530

**Number of Observations: 500**

**Number of Groups: 100**

$$\begin{array}{l} - - = - \quad \overset{\color{red}{\wedge}}{c} \\ - - = - \quad + \overset{\color{red}{\wedge}}{c} \end{array}$$

```
> round(model.matrix(fit) [1:5,],4)
```

```
(Intercept) sp(month)D1(0) sp(month)D2(0) sp(month)C(0).0
1           1           -9.8637           48.6464           0
2           1           -6.9539           24.1783           0
3           1           -2.1141            2.2347           0
4           1            0.0534            0.0014           1
5           1            0.9739            0.4742           1
sp(month)C(0).1 sp(month)C(0).2 sp(month)C(5).2 sp(month)C(-5).2
1           0.0000           0.0000           0           -11.8278
2           0.0000           0.0000           0           -1.9088
3           0.0000           0.0000           0            0.0000
4           0.0534           0.0014           0            0.0000
5           0.9739           0.4742           0            0.0000
```

## Natural spline

Linear in the extreme intervals

*You can create almost every kind of spline used in regression*

```
> spn <- function( x ) gsp( x,  
    knots      = c(-10,-5, 0, 5,10),  
    degree     = c(1, 2, 2, 2, 2, 1),  
    smooth     = c( 1, 1,-1, 1, 1))  
  
> fitn <- lme( sleep ~ spn(month),  
    sim.data, random = ~ 1 | id )  
  
> summary(fitn)  
  
. . .
```

Fixed effects: sleep ~ spn(month)

	Value	Std.Error	DF	t-value	p-value
(Intercept)	7.770115	0.07572201	393	102.61369	0.0000
spn(month)D1(0)	0.007163	0.04256467	393	0.16829	0.8664
spn(month)D2(0)	-0.027267	0.01214689	393	-2.24475	0.0253
spn(month)C(0).0	-2.278971	0.10190952	393	-22.36269	0.0000
spn(month)C(0).1	0.290233	0.06946431	393	4.17816	0.0000
spn(month)C(0).2	-0.013372	0.01882470	393	-0.71035	0.4779
spn(month)C(5).2	0.024200	0.02135186	393	1.13338	0.2577
spn(month)C(-5).2	-0.048604	0.01975742	393	-2.46003	0.0143

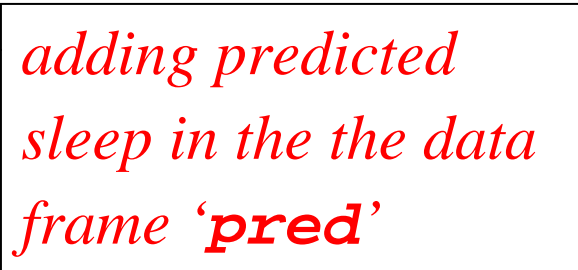
Note: Although the estimable coefficients have the same interpretation for both splines, the natural spline matrix is necessarily different. There are no estimable coefficients attached to the boundary knots.



## Viewing the fitted model:

Create a data frame of values of functional predictors – here just months:

```
> pred <- expand.grid( month = seq(-13,13,.1))2
> pred$sleep <- predict( fitn, pred, level =0)
      # Add predicted sleep to the 'pred' data frame
> head( pred )
  month    sleep
1 -13.0 6.832263
2 -12.9 6.835944
3 -12.8 6.839625
4 -12.7 6.843306
5 -12.6 6.846987
6 -12.5 6.850669
```

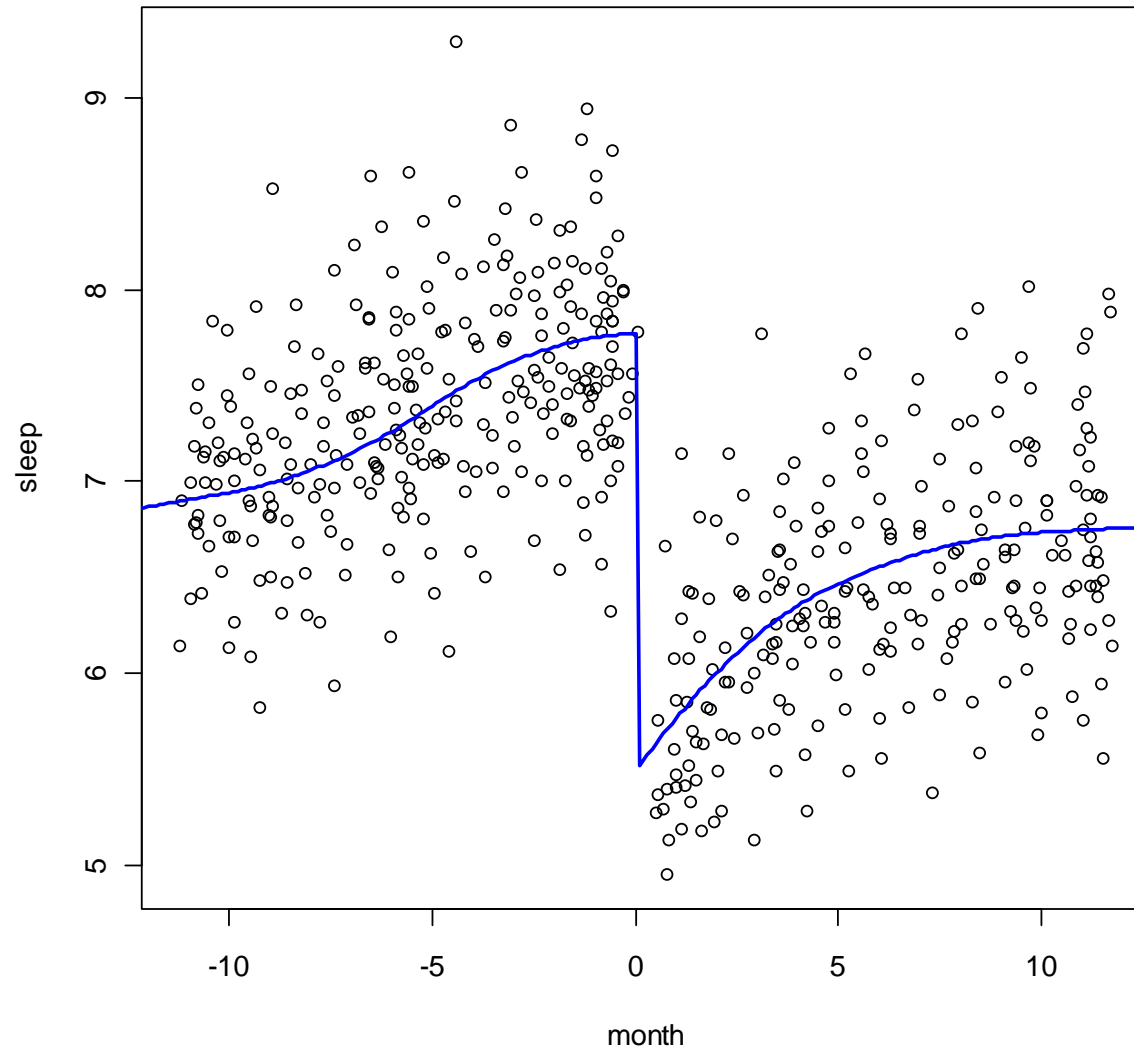


*adding predicted  
sleep in the the data  
frame 'pred'*

---

<sup>2</sup> I used `expand.grid` although it is really needed only for multiple functional predictors

```
> plot( sleep ~ month, sim.data)  
> lines( sleep ~ month, pred, col = 'blue', lwd = 2)
```



# Estimation and testing:

Overall test for effect of time:

```
> wald( fitn, 'month')
```

```
      numDF denDF  F.value p.value  
month      7   393 366.2914 <.00001
```

*'month' is a "regular expression" matching names of coefficients*

Coefficients	Estimate	Std.Error	DF	t-value	p-value
spn(month)D1(0)	0.007163	0.042565	393	0.168293	0.86644
spn(month)D2(0)	-0.027267	0.012147	393	-2.244749	0.02534
spn(month)C(0).0	-2.278971	0.101910	393	-22.362687	<.00001
spn(month)C(0).1	0.290233	0.069464	393	4.178156	0.00004
spn(month)C(0).2	-0.013372	0.018825	393	-0.710346	0.47791
spn(month)C(5).2	0.024200	0.021352	393	1.133384	0.25774
spn(month)C(-5).2	-0.048604	0.019757	393	-2.460034	0.01432

Can we get rid of knots and just use a quadratic?

```
> wald( fitn, 'C')
```

```
  numDF denDF  F.value p.value  
C      5   393 402.4999 <.00001
```

No



Coefficients	Estimate	Std.Error	DF	t-value	p-value
spn(month)C(0).0	-2.278971	0.101910	393	-22.362687	<.00001
spn(month)C(0).1	0.290233	0.069464	393	4.178156	0.00004
spn(month)C(0).2	-0.013372	0.018825	393	-0.710346	0.47791
spn(month)C(5).2	0.024200	0.021352	393	1.133384	0.25774
spn(month)C(-5).2	-0.048604	0.019757	393	-2.460034	0.01432

# Can we get rid of non-zero knots?

```
> wald( fitn, 'C\\(5|C\\(-5')
# '(' is a special character and needs to be 'escaped' to
# have its literal meaning
# '|' means 'or'
      numDF denDF  F.value p.value
C\\(5|C\\(-5      2   393 3.646731 0.02696
```

Coefficients	Estimate	Std.Error	DF	t-value	p-value
spn(month)C(5).2	0.024200	0.021352	393	1.133384	0.25774
spn(month)C(-5).2	-0.048604	0.019757	393	-2.460034	0.01432

## Using a hypothesis matrix:

```
> Lm <- rbind("at 5 mos" = c(0,0,0,0,0,0,1,0),
              "at -5 mos" = c(0,0,0,0,0,0,0,1))
> Lm
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
at 5 mos    0    0    0    0    0    0    1    0
at -5 mos    0    0    0    0    0    0    0    1

> wald( fitn, Lm)
  numDF denDF  F.value p.value
1      2   393 3.646731 0.02696

      Estimate Std.Error  DF   t-value p-value
at 5 mos  0.024200  0.021352 393   1.133384 0.25774
at -5 mos -0.048604  0.019757 393  -2.460034 0.01432
```

# Generating Standard Errors

Wald tests generate estimated standard errors for coefficients, we need to capture them for plotting

Example: plotting the estimated response

L matrix as model matrix evaluated over a range of predictors

```
> Lest <- with( pred, cbind( 1, spn(month)))
> dim(Lest)
[1] 261    8
> head(Lest)
      D1(0) D2(0) C(0).0 C(0).1 C(0).2 C(5).2 C(-5).2
f(-13)   1 -13.0   80      0      0      0      0     -27.5
f(-12.9) 1 -12.9   79      0      0      0      0     -27.0
f(-12.8) 1 -12.8   78      0      0      0      0     -26.5
f(-12.7) 1 -12.7   77      0      0      0      0     -26.0
f(-12.6) 1 -12.6   76      0      0      0      0     -25.5
f(-12.5) 1 -12.5   75      0      0      0      0     -25.0
```

```

> head( pred )
  month      sleep
1 -13.0  6.832263
2 -12.9  6.835944
3 -12.8  6.839625
4 -12.7  6.843306
5 -12.6  6.846987
6 -12.5  6.850669

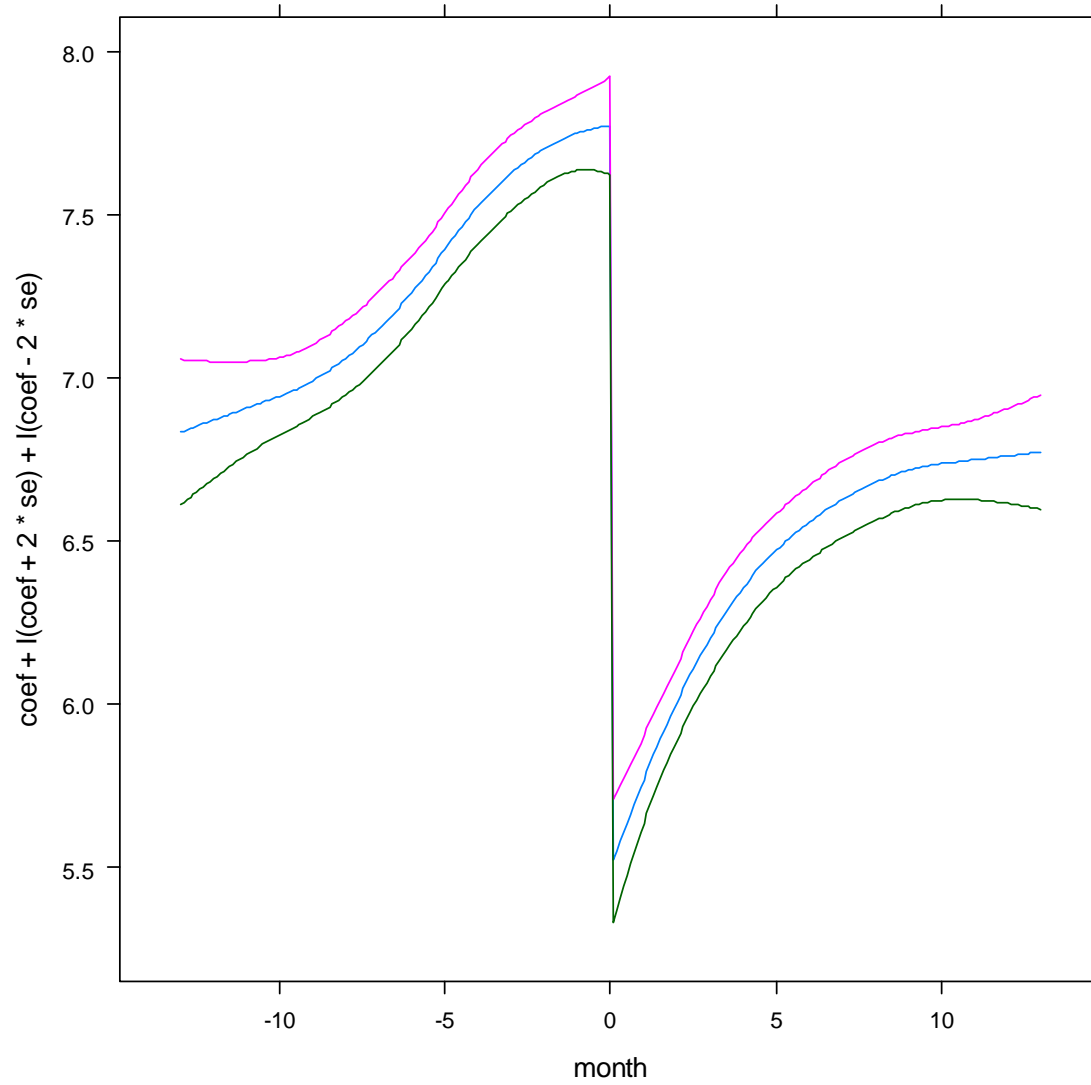
> ww <- as.data.frame(wald(fitn, Lest))
      # capture estimates and SEs
> ww <- cbind( ww, pred)
      # combine back with pred
> head( ww )
      coef          se month      sleep
f(-13)  6.832263 0.1111460 -13.0  6.832263
f(-12.9) 6.835944 0.1089242 -12.9  6.835944
f(-12.8) 6.839625 0.1067182 -12.8  6.839625
f(-12.7) 6.843306 0.1045293 -12.7  6.843306
f(-12.6) 6.846987 0.1023583 -12.6  6.846987
f(-12.5) 6.850669 0.1002066 -12.5  6.850669

```



## Easy plotting:

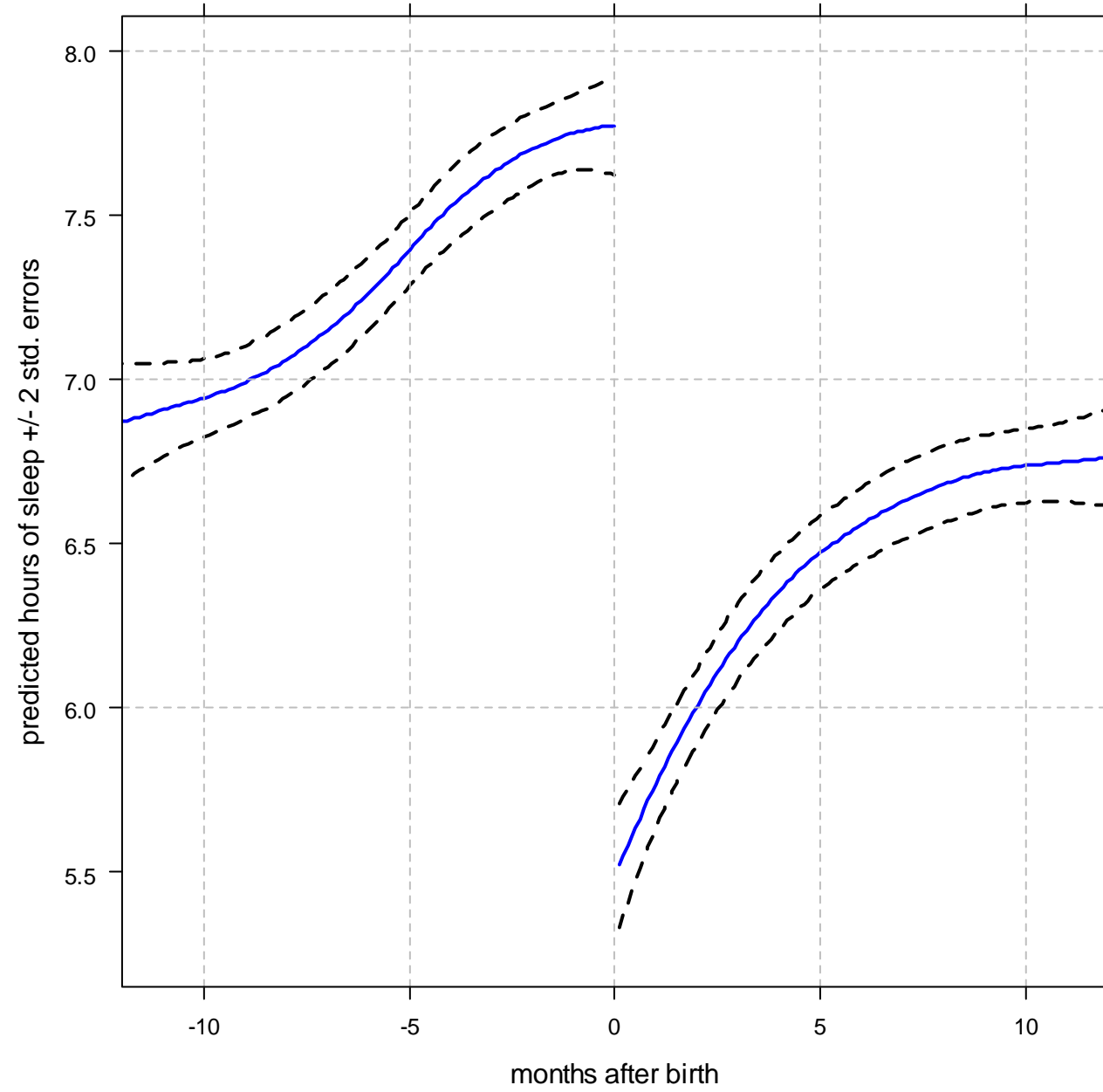
```
> xyplot(coef + I(coef + 2*se) + I(coef - 2*se) ~  
month, ww, type = 'l')
```



Fancier plotting:

```
> td( col = c('blue','black','black'),
      lty = c( 1,2,2), lwd = 2)

> xyplot( coef + I(coef + 2*se) + I(coef - 2*se) ~
          month,
          rbind(ww[1:131,],NA,ww[132:261,]),
          type = 'l',
          panel = function(x,y,...) {
            panel.xyplot( x, y, ...)
            panel.abline( v = c(-10,-5,0,5,10),
                          col = 'gray', lty = 2)
            panel.abline( h = seq(0,10,1),
                          col = 'gray', lty = 2)
          },
          xlim = c(-12,12),
          ylab =
            'predicted hours of sleep +/- 2 std. errors',
          xlab = "months after birth")
```



# Inference concerning a spline

All fixed effects coefficients:

```
> wald( fitn )
      numDF denDF  F.value p.value
      8     393 2688.578 <.00001

Coefficients      Estimate Std.Error  DF    t-value p-value
(Intercept)      7.770115  0.075722 393  102.613693 <.00001
spn(month)D1(0)  0.007163  0.042565 393   0.168293 0.86644
spn(month)D2(0) -0.027267  0.012147 393  -2.244749 0.02534
spn(month)C(0).0 -2.278971  0.101910 393 -22.362687 <.00001
spn(month)C(0).1  0.290233  0.069464 393   4.178156 0.00004
spn(month)C(0).2 -0.013372  0.018825 393  -0.710346 0.47791
spn(month)C(5).2  0.024200  0.021352 393   1.133384 0.25774
spn(month)C(-5).2 -0.048604  0.019757 393  -2.460034 0.01432
```

Test saltus at birth (month = 0):

```
> wald( fitn , "C\\(0.*0$")
      numDF denDF  F.value p.value
C\\(0.*0$      1     393 500.0898 <.00001

Coefficients      Estimate Std.Error  DF    t-value p-value
spn(month)C(0).0 -2.278971  0.10191 393 -22.36269 <.00001
```

## Better labelled:

```
> wald( fitn , list("discontinuity at 0" = "C\\(0.*0$"))
              numDF denDF  F.value p.value
discontinuity at 0      1   393 500.0898 <.00001
```

## Operating on splines: `sc` function

```
sc( spn, x, D, type)
```

generates a hypothesis matrix to estimate the  $D^{\text{th}}$  power coefficient of the spline `spn` at the points given by `x`. ‘`type`’ only has an effect when the value of `x` corresponds to a knot. In that case, `type` determines whether the evaluation is to the left, to the right or across the knot.

sc parameters:		
D	0	value of spline
	1	first derivative
	2,...	quadratic,... terms
type	0	limit from the left of knot
	1	limit from the right of knot
	2	saltus: limit from the right minus limit from the left

## Example:

Estimate value, slope and quadratic component at -1 months:

```
> sc(spn, c(-1,-1,-1), D = c(1,2,3) )
      D1(0) D2(0) C(0).0 C(0).1 C(0).2 C(5).2 C(-5).2
D1(-1)    1   -1     0     0     0     0     0
D2(-1)    0    1     0     0     0     0     0
D3(-1)    0    0     0     0     0     0     0
```

```
> wald( fitn, cbind(c(1,0,0,0),
+                  sc(spn, c(-1,-1,-1,-1), D = c(0,1,2,3) )))
```

```
  numDF denDF  F.value p.value
1      3   393 6937.729 <.00001
```

```
      Estimate Std.Error  DF    t-value p-value
g(-1)  7.749318  0.057332 393 135.165234 <.00001
D1(-1)  0.034430  0.030741 393   1.120008 0.26339
D2(-1) -0.027267  0.012147 393  -2.244749 0.02534
D3(-1)  0.000000  0.000000 Inf      NaN     NaN
```

Using the sc function to test a saltus (type = 2):

```
> Lm <- cbind( 0, sc( spn, 0, D = 0, type = 2))
```

```
> Lm
```

	D1(0)	D2(0)	C(0).0	C(0).1	C(0).2	C(5).2	C(-5).2
g(0+)-g(0-)	0	0	1	0	0	0	0

```
> wald( fitn, Lm )
```

	numDF	denDF	F.value	p.value
1	1	393	500.0898	<.00001

	Estimate	Std.Error	DF	t-value	p-value
g(0+)-g(0-)	-2.278971	0.10191	393	-22.36269	<.00001

```
> head( model.matrix( fitn))
```

	(Intercept)	spn(month)D1(0)	spn(month)D2(0)	spn(month)C(0).0
1	1	-9.86370968	48.646384300	0
2	1	-6.95388563	24.178262681	0
3	1	-2.11407625	2.234659188	0
4	1	0.05344575	0.001428224	1
5	1	0.97390029	0.474240891	1
6	1	-7.66649560	29.387577401	0

	spn(month)C(0).1	spn(month)C(0).2	spn(month)C(5).2	spn(month)C(-5).2
1	0.00000000	0.000000000	0.000000e+00	-11.827836
2	0.00000000	0.000000000	0.000000e+00	-1.908835
3	0.00000000	0.000000000	0.000000e+00	0.000000
4	0.05344575	0.001428224	-1.441111e-18	0.000000
5	0.97390029	0.474240891	1.717766e-16	0.000000
6	0.00000000	0.000000000	0.000000e+00	-3.555099

```
> L.level <- cbind( 1, spn( c(-10, 10)))
```

```
> L.level
```

	D1(0)	D2(0)	C(0).0	C(0).1	C(0).2	C(5).2	C(-5).2	
f(-10)	1	-10	50	0	0	0	0.0	-12.5
f(10)	1	10	50	1	10	50	12.5	0.0

```
> wald( fitn, L.level )
```

	numDF	denDF	F.value	p.value
1	2	393	8216.843	<.00001

	Estimate	Std.Error	DF	t-value	p-value	Lower 0.95	Upper 0.95
f(-10)	6.942695	0.058820	393	118.0330	<.00001	6.827054	7.058337
f(10)	6.735666	0.055838	393	120.6283	<.00001	6.625887	6.845445



```

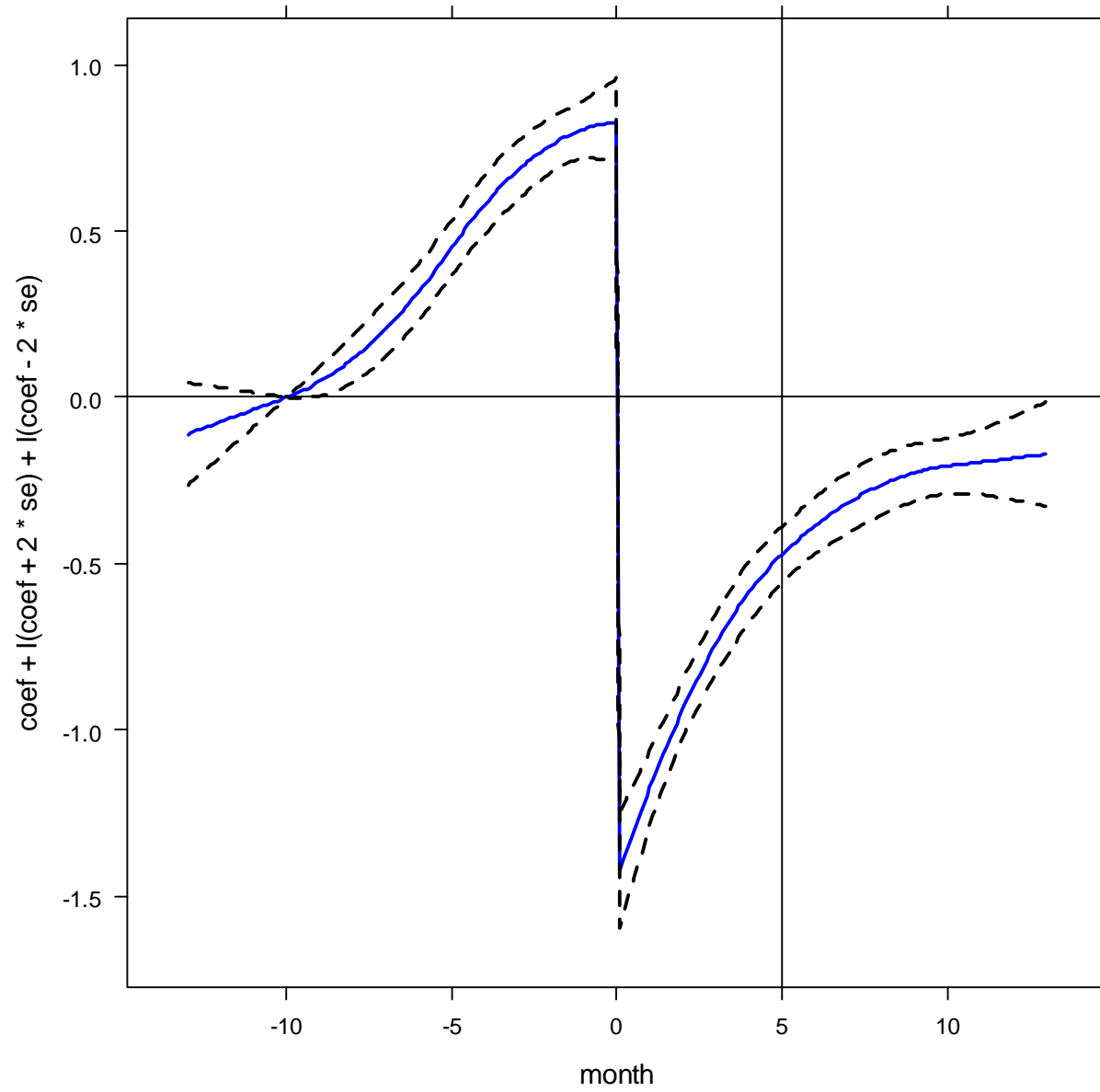
> wald( fitn, L.level[2,] - L.level[1,] )
  numDF denDF F.value p.value
1      1   393 24.4368 <.00001

      Estimate Std.Error  DF   t-value p-value Lower 0.95 Upper 0.95
Larg -0.20703   0.04188 393 -4.943359 <.00001 -0.289367 -0.124692

> ?sc
> Lslope <- with( pred, cbind( 0 , sc( spn, month, D = 1)))
> ws <- as.data.frame(wald(fitn,Lslope))
> head(ws)
           coef          se
D1(-13)  0.03681075 0.02576518
D1(-12.9) 0.03681075 0.02576518
D1(-12.8) 0.03681075 0.02576518
D1(-12.7) 0.03681075 0.02576518
D1(-12.6) 0.03681075 0.02576518
D1(-12.5) 0.03681075 0.02576518
> ws <- cbind(ws, pred)

```

```
> xyplot( coef + I(coef + 2*se)+I(coef - 2*se) ~ month,  
          ws, type = 'l')
```

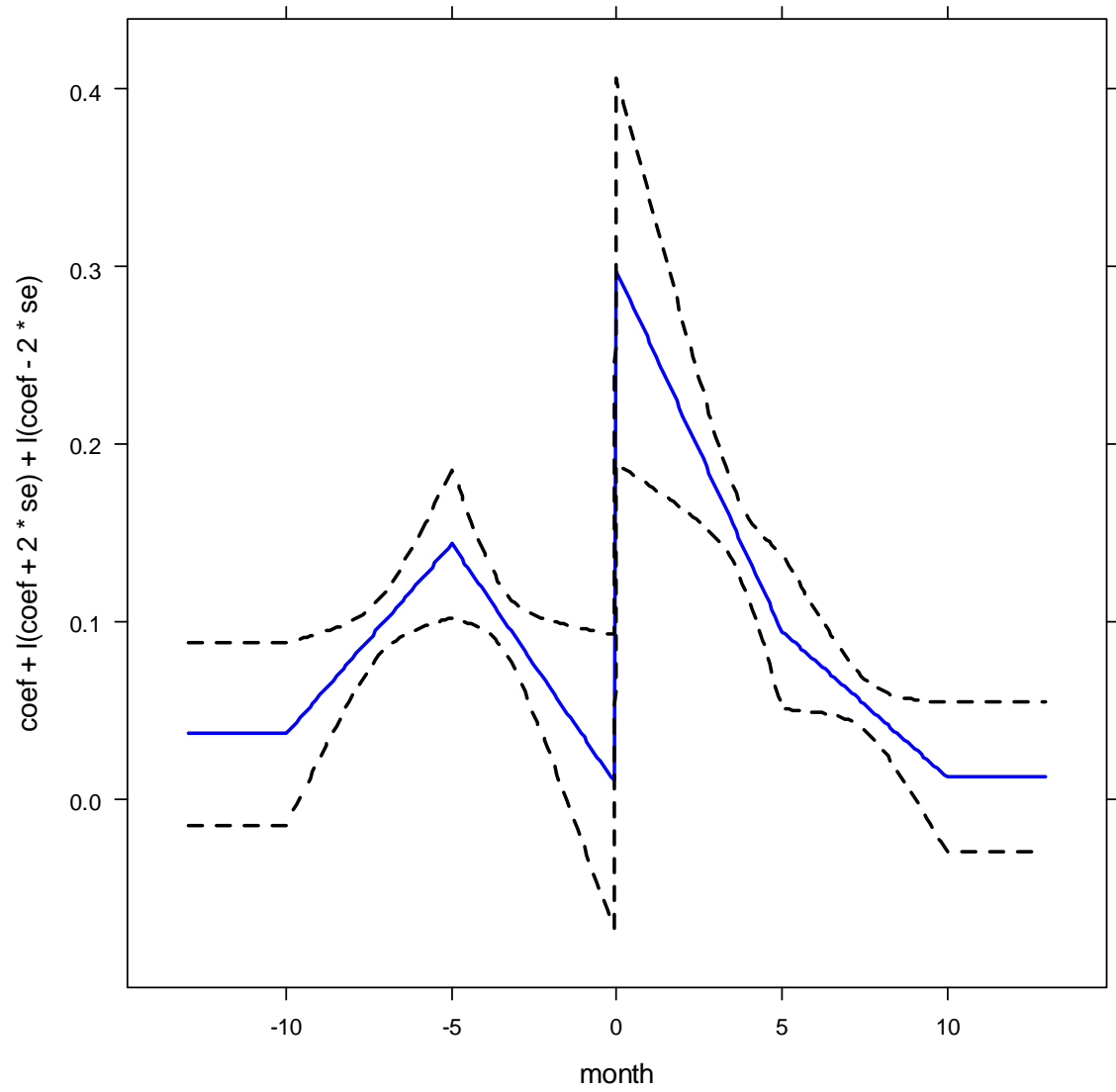


```

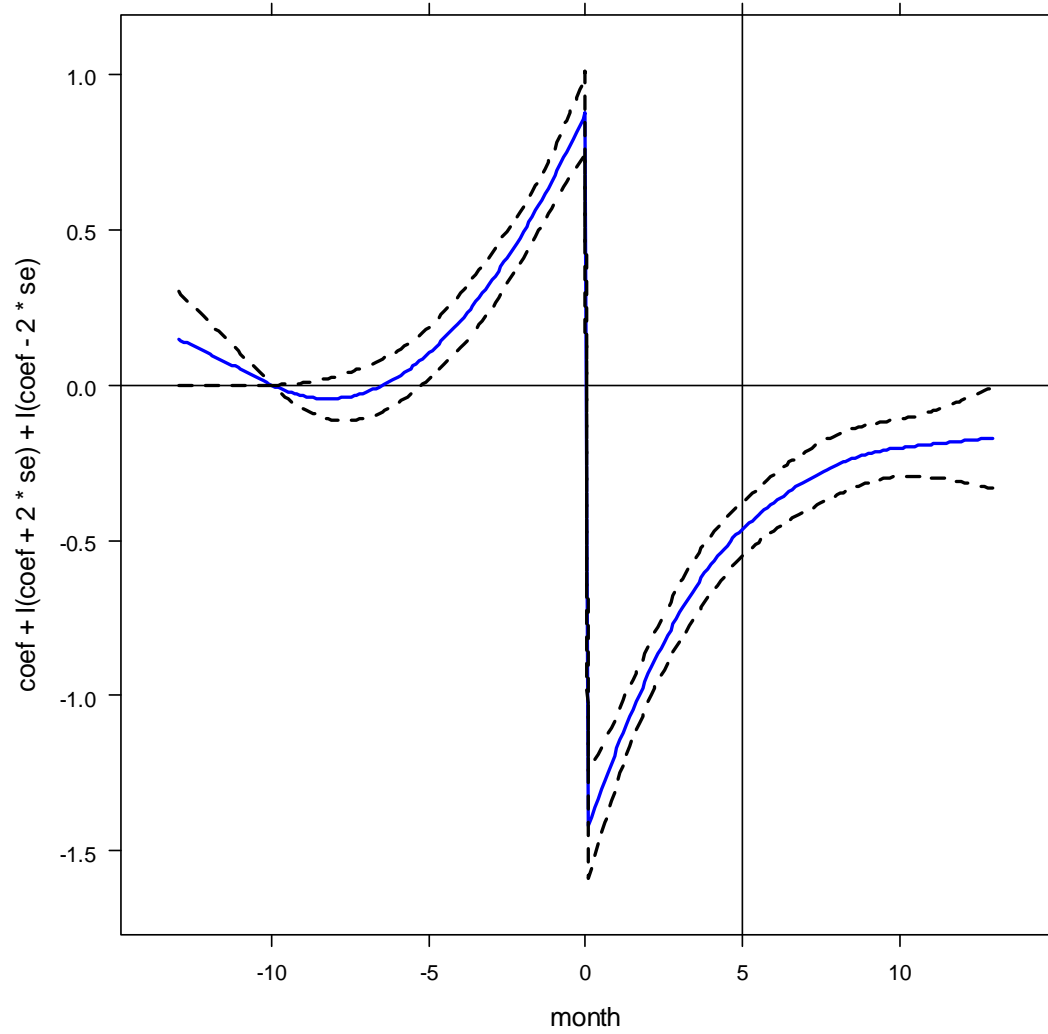
> head( Lest )
      D1(0) D2(0) C(0).0 C(0).1 C(0).2 C(5).2 C(-5).2
f(-13)  1 -13.0   80     0     0     0     0    -27.5
f(-12.9) 1 -12.9   79     0     0     0     0    -27.0
f(-12.8) 1 -12.8   78     0     0     0     0    -26.5
f(-12.7) 1 -12.7   77     0     0     0     0    -26.0
f(-12.6) 1 -12.6   76     0     0     0     0    -25.5
f(-12.5) 1 -12.5   75     0     0     0     0    -25.0
> L.m10 <- cbind( 1, spn(-10))
> L.m10
      D1(0) D2(0) C(0).0 C(0).1 C(0).2 C(5).2 C(-5).2
f(-10)  1  -10   50     0     0     0     0    -12.5
> Ldiff.m10 <- Lest - L.m10[rep(1,nrow(Lpred)),]
> wd <- as.data.frame( wald( fitn, Ldiff.m10))
> wd <- cbind(wd, pred)
> wdr <- as.data.frame( wald( fitnr, Ldiff.m10))
> wdr <- cbind(wdr, pred)

```

```
> xyplot( coef + I(coef + 2*se)+I(coef - 2*se)
~ month, wd, type = 'l',
abline = list(h=0,v=5))
```



```
> xyplot( coef + I(coef + 2*se)+I(coef - 2*se) ~  
          month, wdr, type = 'l',  
          abline = list(h=0, v=5))
```



## Caution:

Since `gsp` uses raw parametrization (in contrast with b-splines, for example), you should center and rescale the predictor. A range within -10 to 10 for a spline with no power higher than cubic should be adequate.