

Using parametric splines and Fourier series for seasonal effects

MATH 4939

2026-03-05

- Using parametric splines for piece-wise polynomial curves
- and Fourier series for seasonal effects

Data set simulates data from Statistics Canada NPHS from 1994 to 2011. Participants were surveyed every 2 years for up to 7 occasions.

Some participants happened to give birth during the study but since data was collected every two years there was little data on individual longitudinal sleep patterns before and after birth.

However, using mixed models with a parametric model for sleep behaviour before and after birth, it's possible to 'stitch' trajectories together to get a picture of individual predicted sleep trajectories.

```
library(spida2)
library(nlme)
```

This is a modified version of nlme that includes a pdInd class. See ?pdInd

It can be installed with: `remotes::install_github('gmonette/nlme')`

To suppress this message load nlme with `'suppressPackageStartupMessages(library(nlme))'`

Attaching package: 'nlme'

The following object is masked from 'package:spida2':

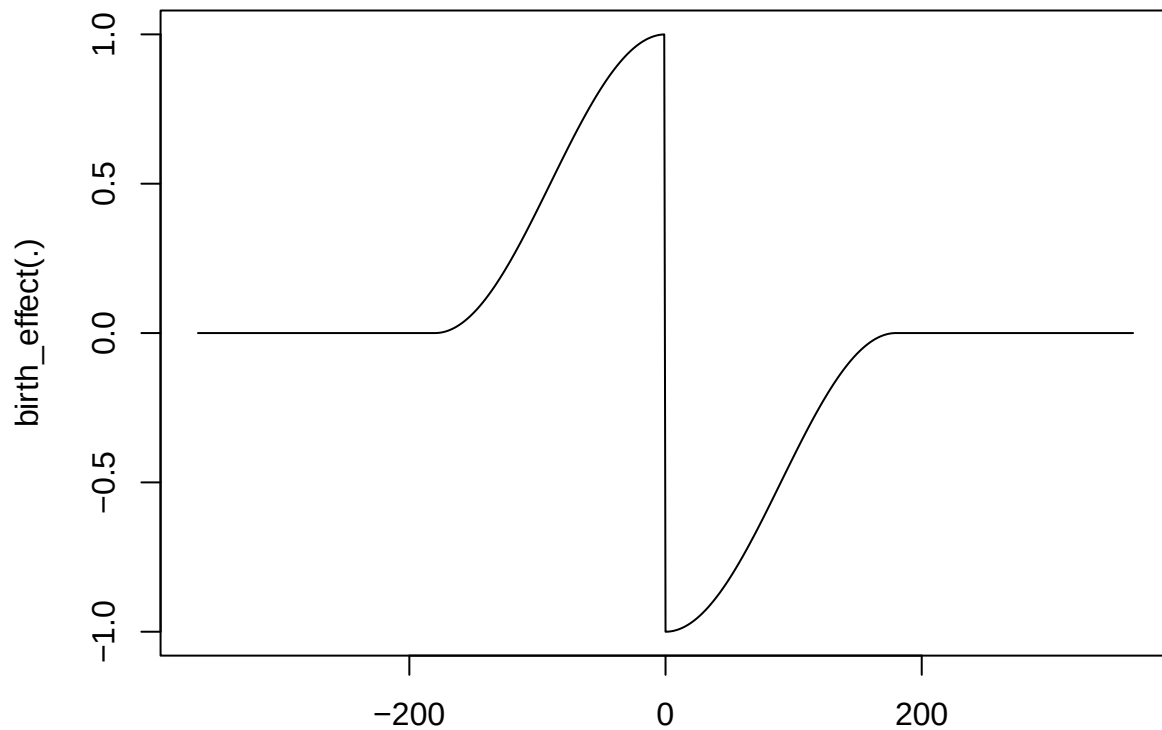
`getData`

```
library(latticeExtra)
```

Loading required package: lattice

Hypothetical perinatal 'birth effect' on maternal sleep relative to days before and after birth.

```
birth_effect <- function(d, plus = 1, minus = 1) {  
  ifelse(d < -180, 0,  
    ifelse(d < 0, plus * (.5 - .5 * cos(pi*(d+180)/180)),  
      ifelse(d < 180, - minus * (.5 - .5 * cos(pi*(d+180)/180)), 0)  
    )  
  )  
}  
  
# test  
seq(-365,365) %>% plot(., birth_effect(.), type = 'l')
```



Generate a data set

Note that many women in the NPHS gave birth more than once. Here there is only one birth recorded per person.

```

# sample(100000, 1)
{
  set.seed(4728)
  Nid <- 1000    # number of subjects
  Nobs <- 7      # observations per subject

  expand.grid(id = 1:Nid, obs = 1:Nobs) %>% # basic skeleton for data set
    within(
      {

        # date id registered
        reg_date <- sample(Nobs * 365, Nid, replace = TRUE)[id] # generating one value per id

        # dates id observed (approx every 2 years)
        date <- reg_date + obs*2*365 + sample(365, length(id), replace = TRUE) # generating one val

        birth_date <- reg_date + sample(365*14, Nid, replace = TRUE)[id] # date giving birth

        ..plus <- runif(Nid)[id] # extra sleep pre birth
        ..minus <- runif(Nid)[id] # less sleep after birth
        ..birth_effect <- birth_effect( date - birth_date, ..plus, ..minus)

        ..seasonal <- .5 * cos(2*pi*(date-30)/365)
        ..sd_between <- 1
        ..sd_within <- .5
      }
    )
}

```

```

sleep <- 8 + ..sd_between * rnorm(Nid)[id] + ..sd_within * rnorm(id) +
  ..birth_effect + ..seasonal

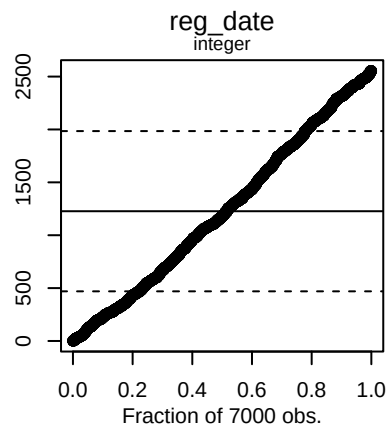
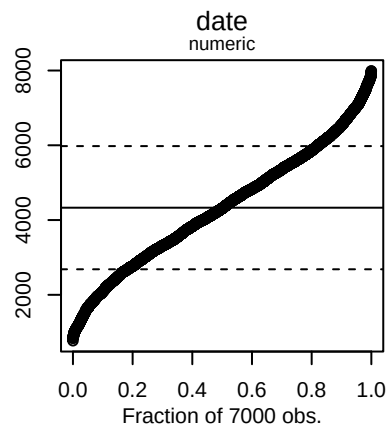
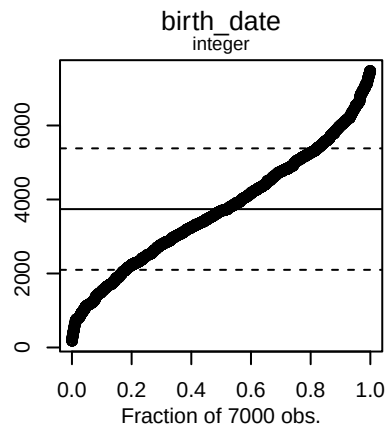
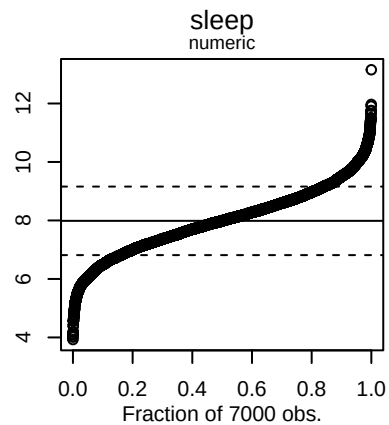
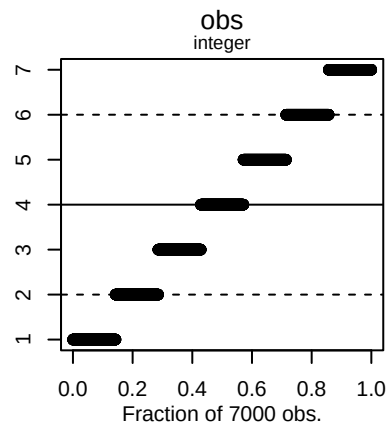
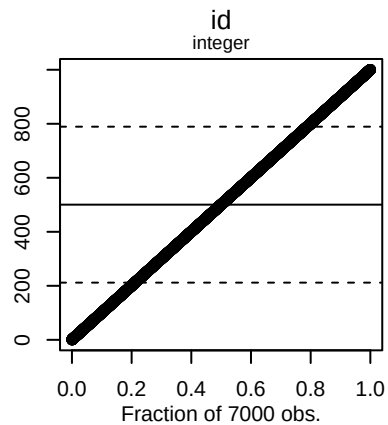
..plus <- ..minus <- ..birth_effect <- ..seasonal <- ..sd_between <- ..sd_within <- NULL

}
) %>%
sortdf(~id/date)-> dd
}
head(dd)

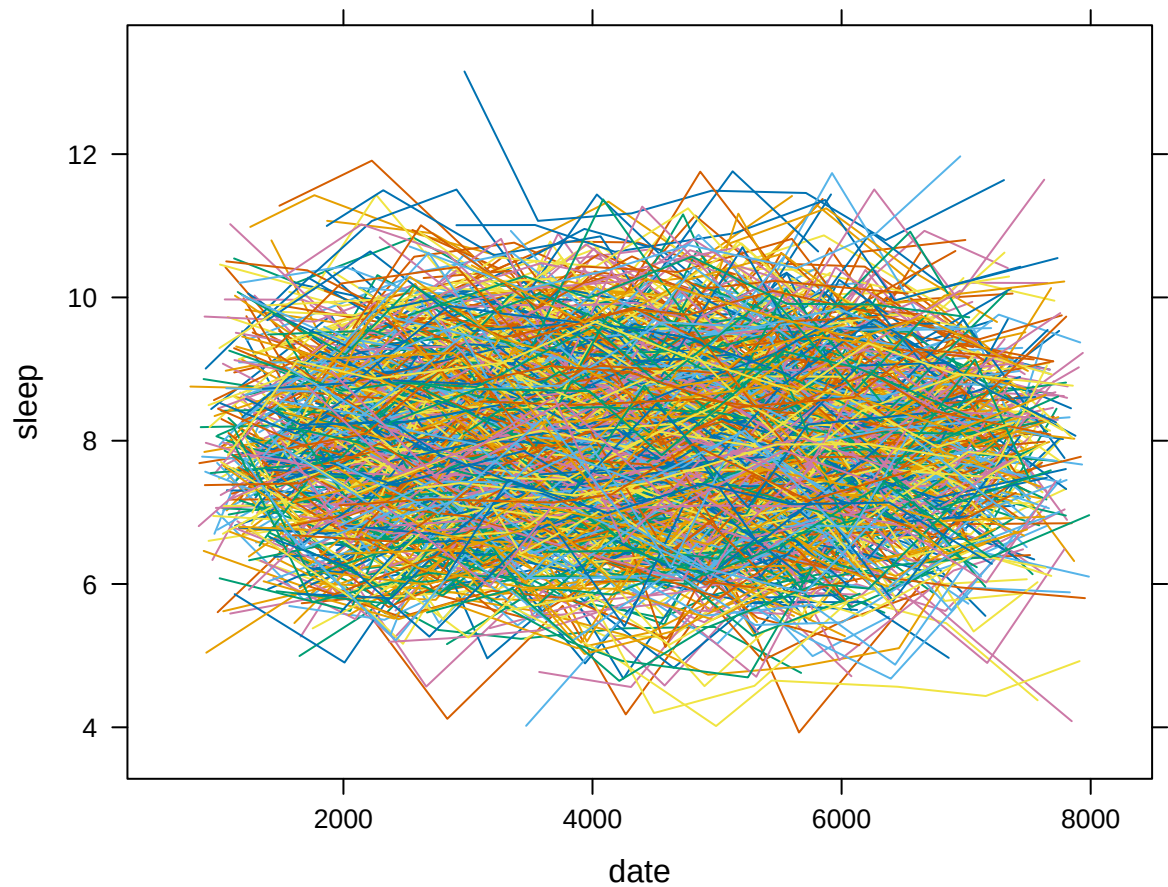
```

	id	obs	sleep	birth_date	date	reg_date
1	1	1	7.088871	5298	1197	288
1001	1	2	7.336316	5298	1891	288
2001	1	3	7.909879	5298	2655	288
3001	1	4	7.489981	5298	3441	288
4001	1	5	7.087731	5298	4084	288
5001	1	6	6.920117	5298	4893	288

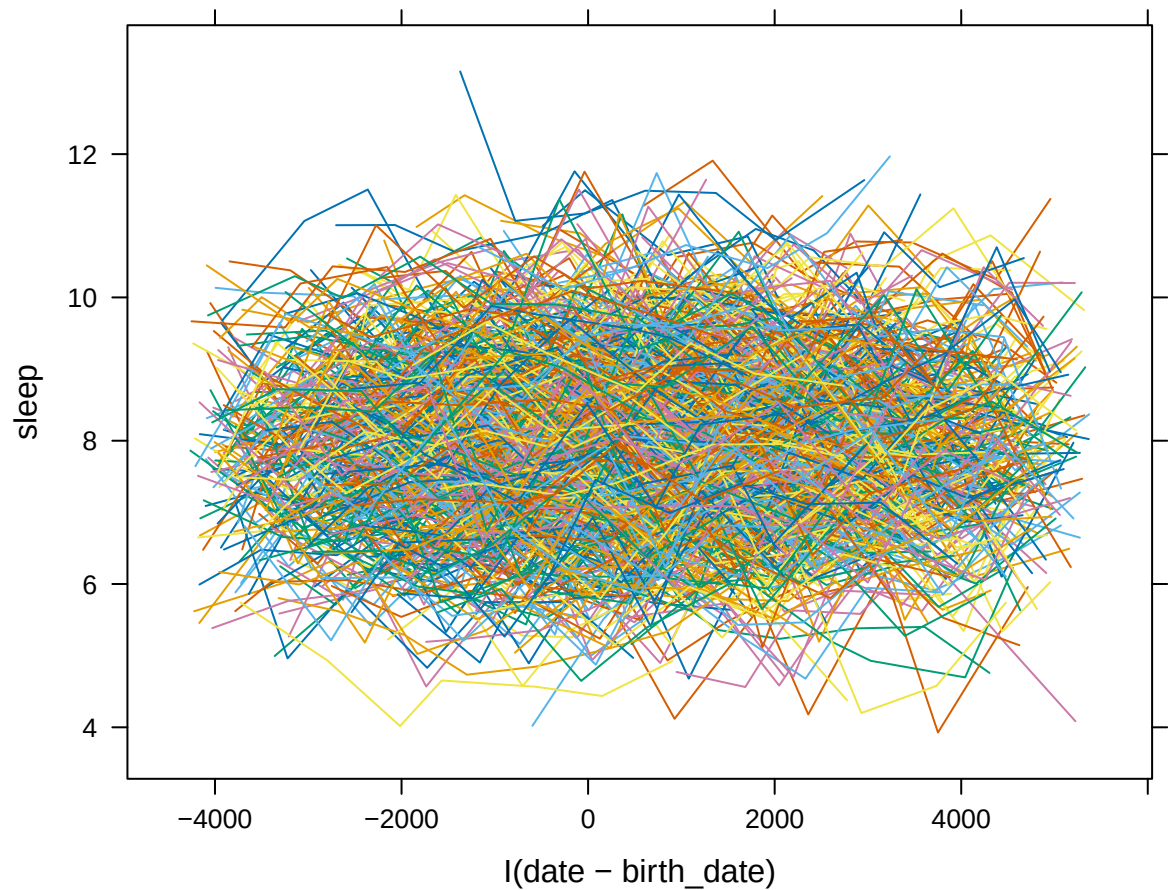
```
xqplot(dd)
```



```
xyplot(sleep ~ date, dd, groups = id, type = 'l')
```



```
xyplot(sleep ~ I(date-birth_date), dd, groups = id, type = 'l')
```



Note: one observation every two years on each person

Between-person and within-person variation in sleep

```
fit <- lme(sleep ~ 1, dd, random = ~1 | id)
summary(fit)
```

Linear mixed-effects model fit by REML

Data: dd

	AIC	BIC	logLik
	16048.53	16069.09	-8021.267

Random effects:

Formula: ~1 | id

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

StdDev:	0.9974714	0.6156731
---------	-----------	-----------

Fixed effects: sleep ~ 1

	Value	Std.Error	DF	t-value	p-value
(Intercept)	7.987087	0.03238981	6000	246.5926	0

Standardized Within-Group Residuals:

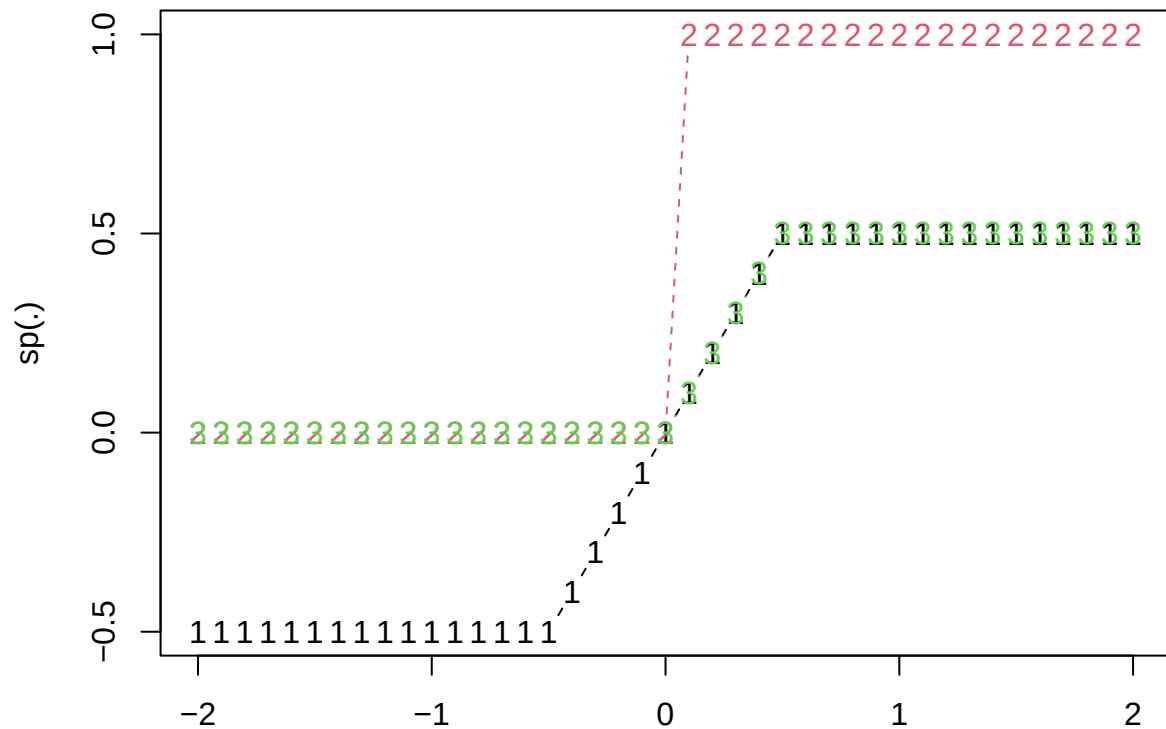
	Min	Q1	Med	Q3	Max
	-3.85320384	-0.64160961	0.00633589	0.63204103	3.68567783

Number of Observations: 7000

Number of Groups: 1000

define a parametric spline using years as unit to avoid large numbers

```
sp <- function(y) {  
  gsp(y, knots = c(-.5,0,.5), degree = c(0,1,1,0), c(0, -1, 0))  
}  
  
seq(-2,2,.1) %>% matplot(., sp(.), type = 'b')
```



```
sp(seq(-2,2,.1))
```

```

      D1(0) C(0).0 C(0).1
[1,] -0.5    0    0.0

```

[2,]	-0.5	0	0.0
[3,]	-0.5	0	0.0
[4,]	-0.5	0	0.0
[5,]	-0.5	0	0.0
[6,]	-0.5	0	0.0
[7,]	-0.5	0	0.0
[8,]	-0.5	0	0.0
[9,]	-0.5	0	0.0
[10,]	-0.5	0	0.0
[11,]	-0.5	0	0.0
[12,]	-0.5	0	0.0
[13,]	-0.5	0	0.0
[14,]	-0.5	0	0.0
[15,]	-0.5	0	0.0
[16,]	-0.5	0	0.0
[17,]	-0.4	0	0.0
[18,]	-0.3	0	0.0
[19,]	-0.2	0	0.0
[20,]	-0.1	0	0.0
[21,]	0.0	0	0.0
[22,]	0.1	1	0.1
[23,]	0.2	1	0.2
[24,]	0.3	1	0.3
[25,]	0.4	1	0.4
[26,]	0.5	1	0.5

[27,]	0.5	1	0.5
[28,]	0.5	1	0.5
[29,]	0.5	1	0.5
[30,]	0.5	1	0.5
[31,]	0.5	1	0.5
[32,]	0.5	1	0.5
[33,]	0.5	1	0.5
[34,]	0.5	1	0.5
[35,]	0.5	1	0.5
[36,]	0.5	1	0.5
[37,]	0.5	1	0.5
[38,]	0.5	1	0.5
[39,]	0.5	1	0.5
[40,]	0.5	1	0.5
[41,]	0.5	1	0.5

Use years as time units

```
dd <- within(dd,
  {
    datey <- date / 365
    birthy <- birth_date / 365
  })
```

```
fit <- lme(sleep ~ sp(datey - birthy) , dd, random = ~ 1 | id)
summary(fit)
```

Linear mixed-effects model fit by REML

Data: dd

AIC	BIC	logLik
15985.45	16026.57	-7986.724

Random effects:

Formula: ~1 | id

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

StdDev:	0.9970469	0.6118624
---------	-----------	-----------

Fixed effects: sleep ~ sp(datey - birthy)

	Value	Std.Error	DF	t-value	p-value
(Intercept)	8.465813	0.08105925	5997	104.43982	0.0000
sp(datey - birthy)D1(0)	0.987491	0.15450552	5997	6.39130	0.0000
sp(datey - birthy)C(0).0	-0.985077	0.11077282	5997	-8.89277	0.0000
sp(datey - birthy)C(0).1	0.040963	0.22128064	5997	0.18512	0.8531

Standardized Within-Group Residuals:

Min	Q1	Med	Q3	Max
-3.877379606	-0.646902798	0.002500854	0.631484893	3.660662143

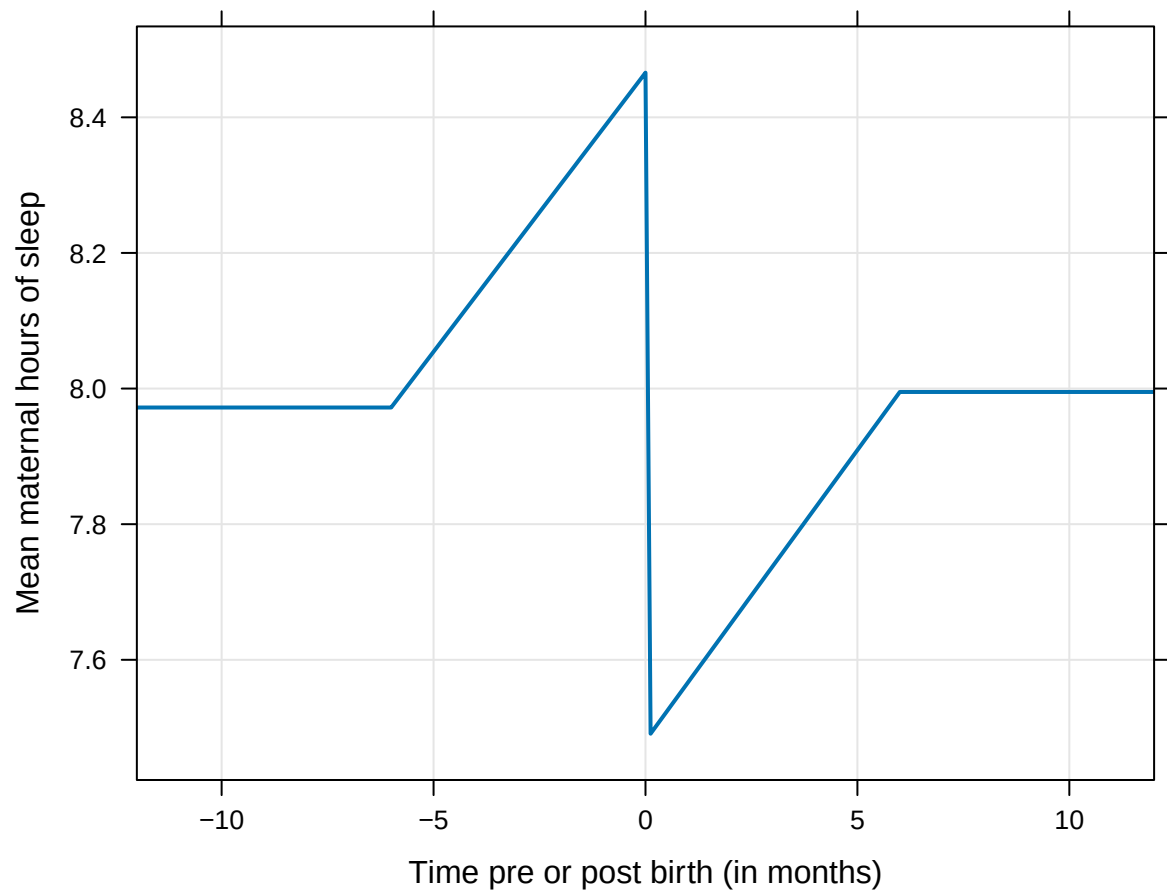
Number of Observations: 7000

Number of Groups: 1000

Create a prediction data frame to show model prediction

```
pred <- data.frame(datey = seq(-2,2,.01), birthy = 0)
pred$fit <- predict(fit, newdata = pred, level = 0)

xyplot(fit ~ I(12*datey), pred, type = 'l', lwd = 2,
       xlim = c(-12,12),
       ylab = "Mean maternal hours of sleep",
       xlab = "Time pre or post birth (in months)") +
  layer_(panel.grid(h=-1,v=-1))
```



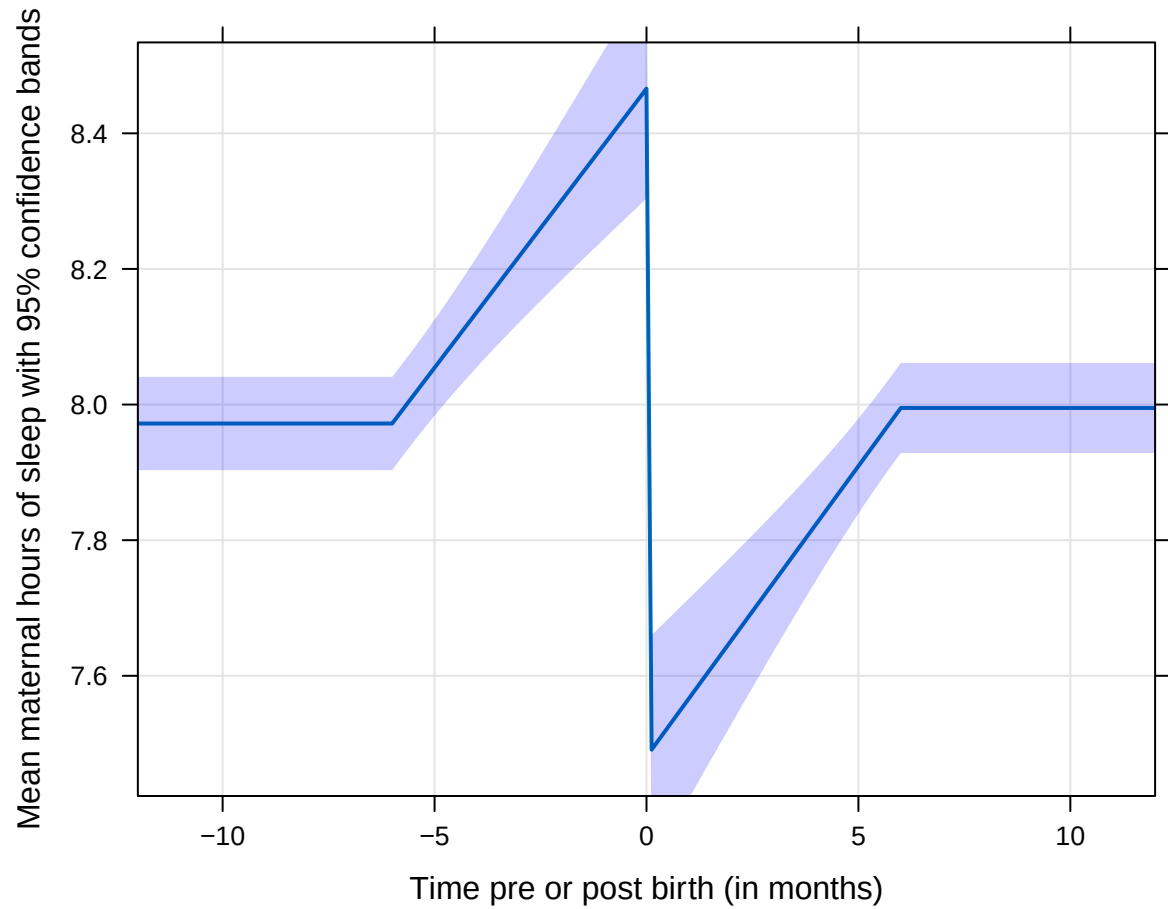
To add error bounds, since 'predict' won't provide them for 'lme' models

```

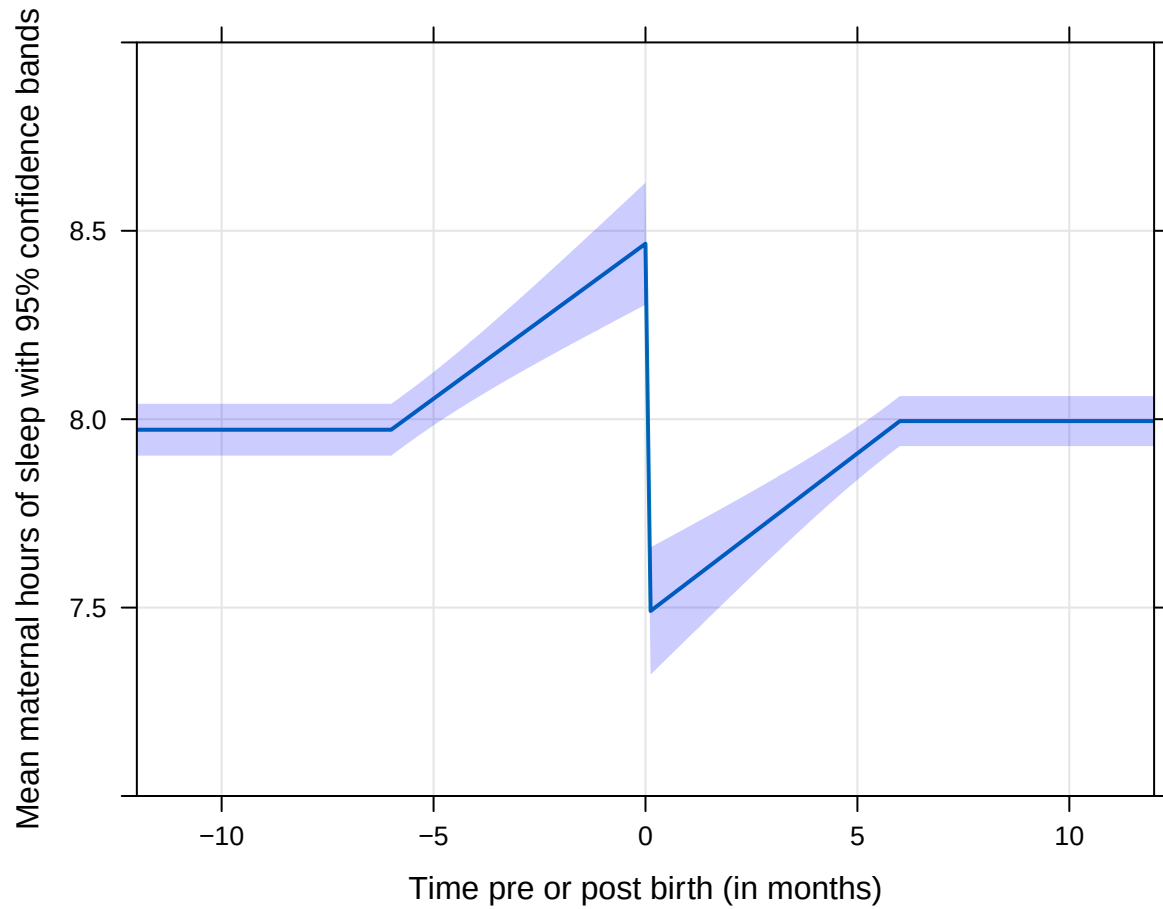
ww <- as.data.frame(wald(fit, pred = pred))

plotbands <- function(ww,...) {
xyplot(coef ~ I(12*datey), ww, type = 'l', lwd = 2,
      xlim = c(-12,12),
      ...,
      lower = ww$L2,           # added for panel.fit
      upper = ww$U2,          # added for panel.fit
      subscripts = T,         # added for panel.fit
      ylab = "Mean maternal hours of sleep with 95% confidence bands",
      xlab = "Time pre or post birth (in months)" ) +
  layer_(panel.grid(h = -1, v = -1)) +
  layer(panel.fit(..., alpha = .2))
}
plotbands(ww)

```



```
plotbands(ww, ylim = c(7,9))
```



Try a different spline

```
sp2 <- function(y) gsp(y, c(-1,-.5, 0, .5, 1), c(0,2,3,3,2,0), c(1,1,-1,1,1))

fit2 <- lme(sleep ~ sp2(datey - birthy) , dd, random = ~ 1 | id)
summary(fit2)
```

Linear mixed-effects model fit by REML

Data: dd

AIC	BIC	logLik
15967.9	16036.42	-7973.948

Random effects:

Formula: ~1 | id

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

StdDev:	0.9971542	0.6118803
---------	-----------	-----------

Fixed effects: sleep ~ sp2(datey - birthy)

	Value	Std.Error	DF	t-value	p-value
(Intercept)	8.55861	0.15489	5993	55.25753	0.0000
sp2(datey - birthy)D1(0)	1.75659	2.02286	5993	0.86837	0.3852
sp2(datey - birthy)D2(0)	1.05499	14.57452	5993	0.07239	0.9423
sp2(datey - birthy)D3(0)	-8.98711	43.99968	5993	-0.20425	0.8382
sp2(datey - birthy)C(0).0	-1.01700	0.21748	5993	-4.67626	0.0000
sp2(datey - birthy)C(0).1	-2.64154	2.94118	5993	-0.89812	0.3692
sp2(datey - birthy)C(0).2	17.97831	21.12807	5993	0.85092	0.3948
sp2(datey - birthy)C(0).3	-60.95171	63.63423	5993	-0.95784	0.3382

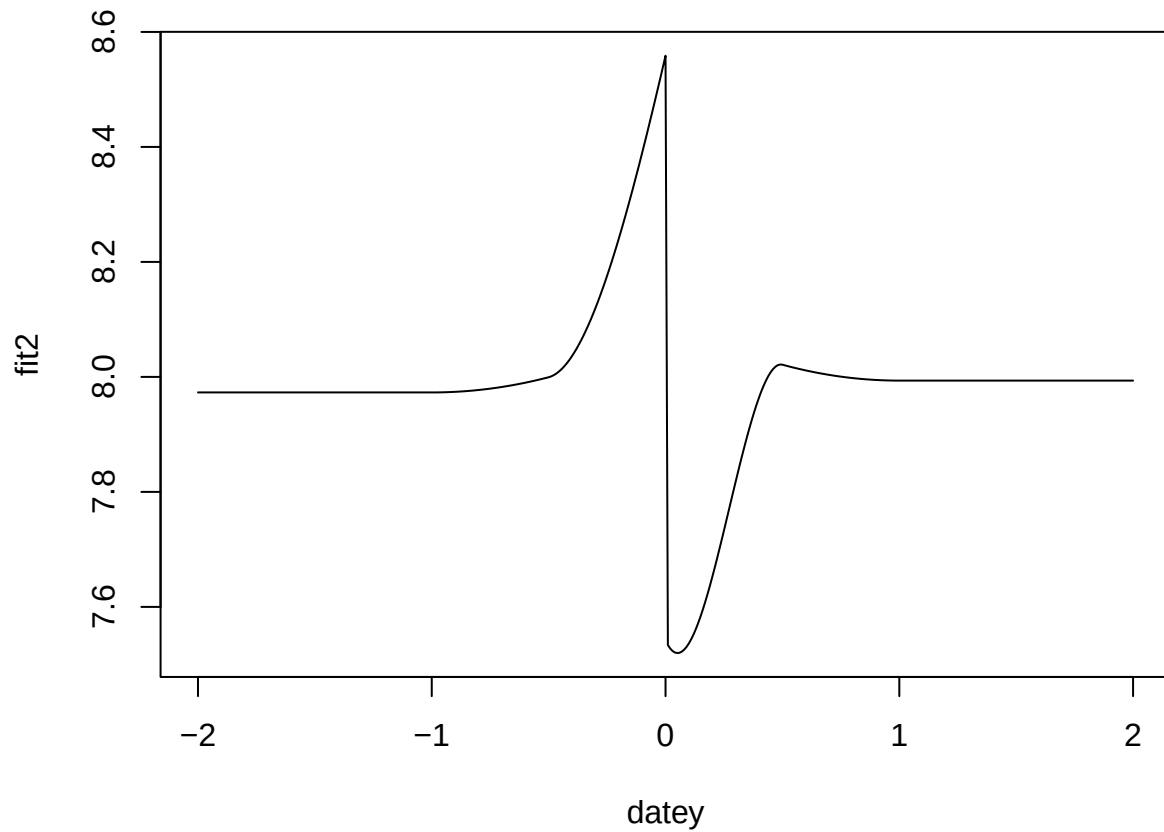
Standardized Within-Group Residuals:

Min	Q1	Med	Q3	Max
-3.877143934	-0.645090676	0.001593104	0.629206506	3.666263366

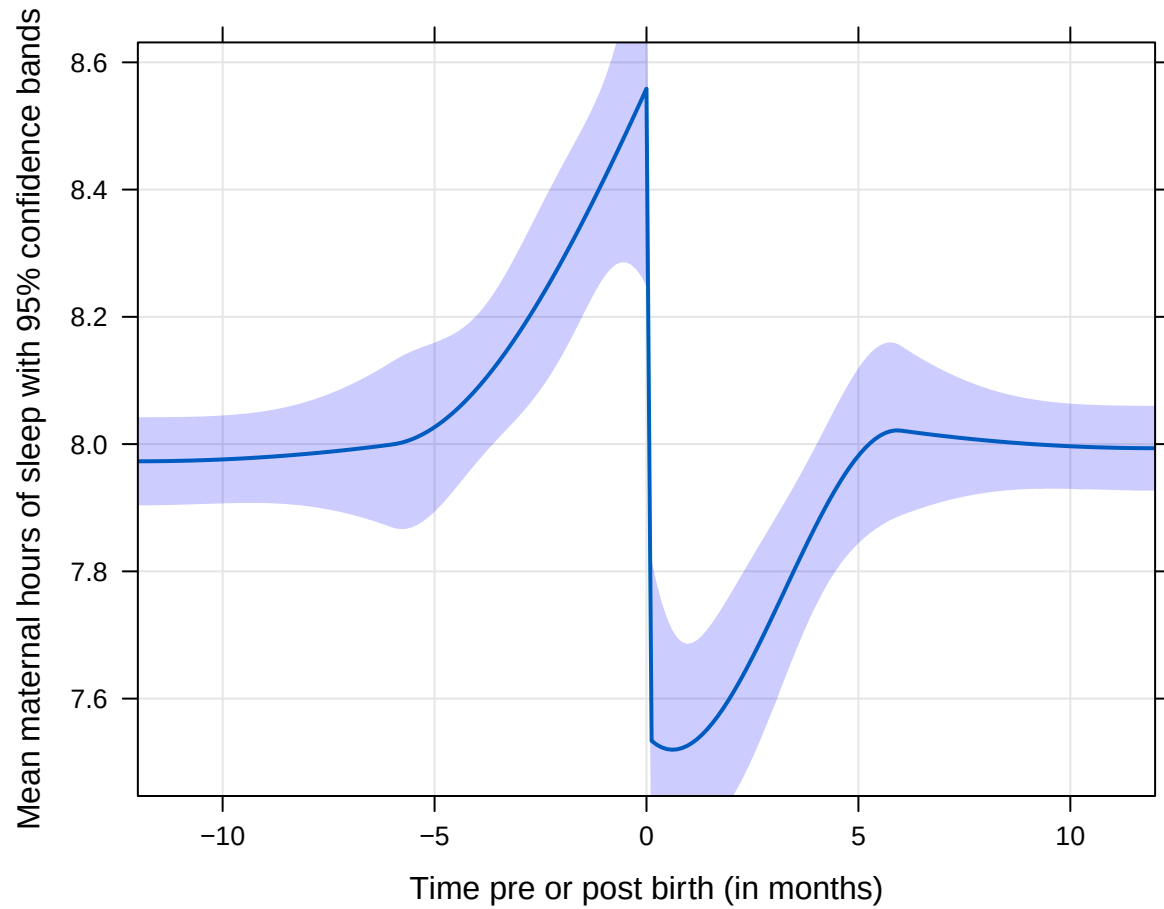
Number of Observations: 7000

Number of Groups: 1000

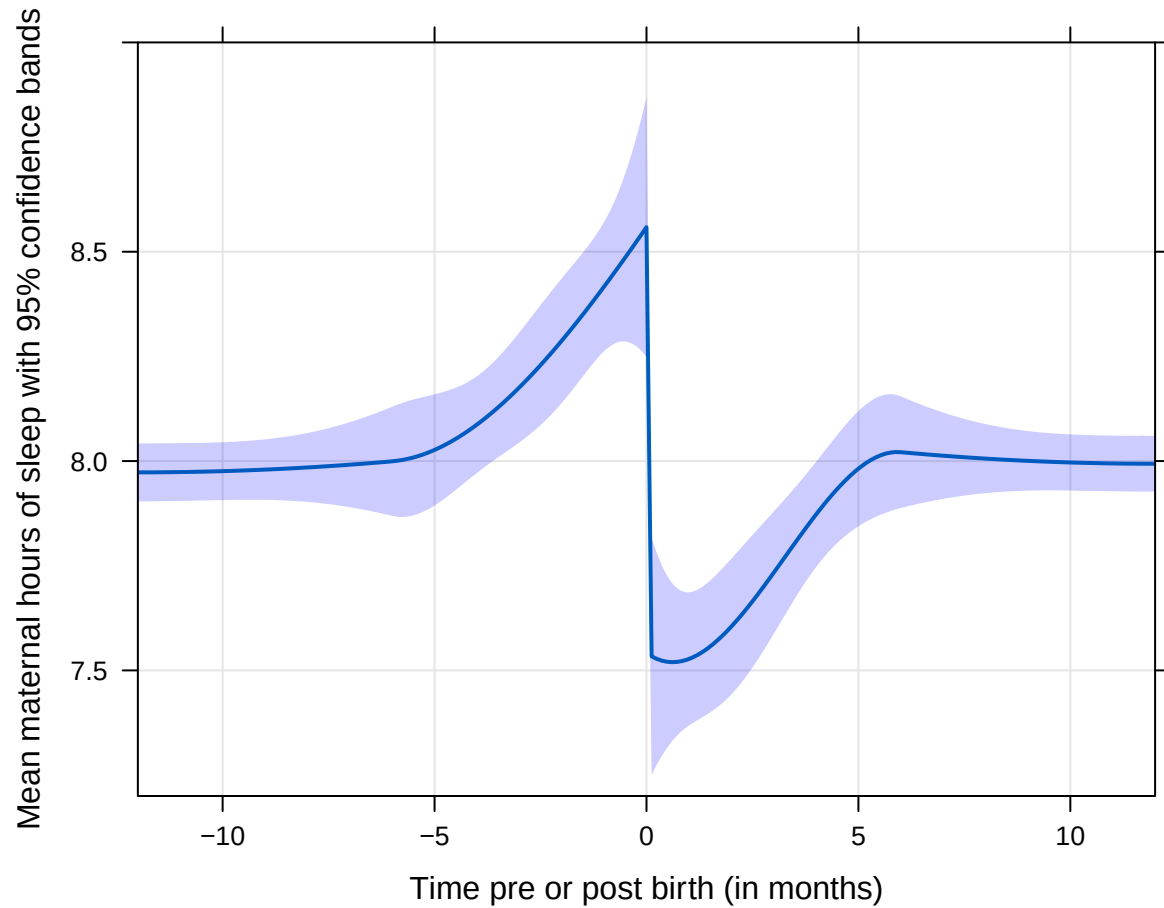
```
pred$fit2 <- predict(fit2, newdata = pred, level = 0)
with(pred, plot(datey, fit2, type = 'l'))
```



```
ww <- as.data.frame(wald(fit2, pred = pred))  
plotbands(ww)
```



```
plotbands(ww, ylim = seq(7.2,9,.2))
```



fit and fit2 have different FE models so we must refit

We can compare these models with AIC or BIC but the p-value should not be interpreted since the neither model is nested in the other

```
anova(update(fit, method = "ML"), update(fit2, method = "ML"))
```

	Model	df	AIC	BIC	logLik	Test
update(fit, method = "ML")	1	6	15970.56	16011.68	-7979.281	
update(fit2, method = "ML")	2	10	15975.12	16043.65	-7977.558	1 vs 2
			L.Ratio	p-value		
update(fit, method = "ML")						
update(fit2, method = "ML")	3.446274		0.4861			

Results: AIC favours the smaller model

The positions of knots can be estimated by trial and error and could be estimated more formally using non-linear models, which we might take up later.

Adding seasonal effects with sin/cos pair harmonics

```
Sin <- function(x) cbind(sin(x), cos(x))
#
fit3 <- lme(sleep ~ sp2(datey - birthy) + Sin(2*pi*datey) , dd, random = ~ 1 | id)
```

We can force *lme* to return an object:

```
fit3 <- lme(sleep ~ sp2(datey - birthy) + Sin(2*pi*datey) , dd, random = ~ 1 | id,
           control = list(returnObject = TRUE))
```

but it's generally better to try an alternative optimizer

```
fit3o <- lme(sleep ~ sp2(datey - birthy) + Sin(2*pi*datey) , dd, random = ~ 1 | id,
            control = list(opt = 'optim', msVerbose = T, verbose = T, returnObject = T))
```

```
initial value 27836.131999
final value 27836.131999
converged
```

with the same result:

```
car::compareCoefs(fit3,fit3o)
```

Calls:

```
1: lme.formula(fixed = sleep ~ sp2(datey - birthy) + Sin(2 * pi * datey),
  data = dd, random = ~1 | id, control = list(returnObject = TRUE))
2: lme.formula(fixed = sleep ~ sp2(datey - birthy) + Sin(2 * pi * datey),
  data = dd, random = ~1 | id, control = list(opt = "optim", msVerbose = T,
  verbose = T, returnObject = T))
```

	Model 1	Model 2
(Intercept)	8.500	8.500
SE	0.129	0.129
sp2(datey - birthy)D1(0)	1.10	1.10
SE	1.67	1.67
sp2(datey - birthy)D2(0)	-0.43	-0.43

SE	12.04	12.04
sp2(datey - birthy)D3(0)	-7.76	-7.76
SE	36.34	36.34
sp2(datey - birthy)C(0).0	-1.06	-1.06
SE	0.18	0.18
sp2(datey - birthy)C(0).1	-0.163	-0.163
SE	2.431	2.431
sp2(datey - birthy)C(0).2	7.62	7.62
SE	17.46	17.46
sp2(datey - birthy)C(0).3	-29.7	-29.7
SE	52.6	52.6
Sin(2 * pi * datey)1	0.25487	0.25487
SE	0.00925	0.00925
Sin(2 * pi * datey)2	0.4213	0.4213
SE	0.0092	0.0092

Also using 'ML' can give convergence:

```
fit3 <- lme(sleep ~ sp2(datey - birthy) + Sin(2*pi*datey) , dd, random = ~ 1 | id, method = 'ML')
```

Compare estimated models

```
summary(fit3)
```

Linear mixed-effects model fit by maximum likelihood

Data: dd

	AIC	BIC	logLik
	13645.35	13727.6	-6810.676

Random effects:

Formula: ~1 | id

(Intercept) Residual

StdDev: 0.9933905 0.504401

Fixed effects: sleep ~ sp2(datey - birthy) + Sin(2 * pi * datey)

	Value	Std.Error	DF	t-value	p-value
(Intercept)	8.500148	0.12914	5991	65.82083	0.0000
sp2(datey - birthy)D1(0)	1.095116	1.67092	5991	0.65540	0.5122
sp2(datey - birthy)D2(0)	-0.429970	12.03848	5991	-0.03572	0.9715
sp2(datey - birthy)D3(0)	-7.757804	36.34333	5991	-0.21346	0.8310
sp2(datey - birthy)C(0).0	-1.062551	0.17968	5991	-5.91358	0.0000
sp2(datey - birthy)C(0).1	-0.162555	2.43072	5991	-0.06688	0.9467
sp2(datey - birthy)C(0).2	7.623029	17.46120	5991	0.43657	0.6624
sp2(datey - birthy)C(0).3	-29.658111	52.58728	5991	-0.56398	0.5728

Sin(2 * pi * datey)1	0.254869	0.00925	5991	27.54351	0.0000
Sin(2 * pi * datey)2	0.421299	0.00920	5991	45.79646	0.0000

Standardized Within-Group Residuals:

Min	Q1	Med	Q3	Max
-3.743602439	-0.633286792	-0.003526948	0.642603214	3.533283785

Number of Observations: 7000

Number of Groups: 1000

`summary(fit3o)`

Linear mixed-effects model fit by REML

Data: dd

AIC	BIC	logLik
13655.9	13738.12	-6815.948

Random effects:

Formula: ~1 | id

(Intercept) Residual

StdDev: 0.9939113 0.504777

Fixed effects: sleep ~ sp2(datey - birthy) + Sin(2 * pi * datey)

	Value	Std.Error	DF	t-value	p-value
(Intercept)	8.500148	0.12914	5991	65.81977	0.0000
sp2(datey - birthy)D1(0)	1.095123	1.67097	5991	0.65538	0.5122

sp2(datey - birthy)D2(0)	-0.429922	12.03884	5991	-0.03571	0.9715
sp2(datey - birthy)D3(0)	-7.757657	36.34439	5991	-0.21345	0.8310
sp2(datey - birthy)C(0).0	-1.062551	0.17969	5991	-5.91340	0.0000
sp2(datey - birthy)C(0).1	-0.162579	2.43079	5991	-0.06688	0.9467
sp2(datey - birthy)C(0).2	7.623092	17.46172	5991	0.43656	0.6624
sp2(datey - birthy)C(0).3	-29.658545	52.58882	5991	-0.56397	0.5728
Sin(2 * pi * datey)1	0.254869	0.00925	5991	27.54273	0.0000
Sin(2 * pi * datey)2	0.421299	0.00920	5991	45.79514	0.0000

Standardized Within-Group Residuals:

	Min	Q1	Med	Q3	Max
	-3.740828341	-0.632796684	-0.003509994	0.642116499	3.530681040

Number of Observations: 7000

Number of Groups: 1000

`getG(fit3)`

Random effects variance covariance matrix

(Intercept)

(Intercept) 0.98682

Standard Deviations: 0.99339

`getG(fit3o)`

Random effects variance covariance matrix

(Intercept)

```
(Intercept)      0.98786
Standard Deviations: 0.99391
```

```
getR(fit3)
```

```
id 1
Conditional variance covariance matrix
      1      2      3      4      5      6      7
1 0.25442 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
2 0.00000 0.25442 0.00000 0.00000 0.00000 0.00000 0.00000
3 0.00000 0.00000 0.25442 0.00000 0.00000 0.00000 0.00000
4 0.00000 0.00000 0.00000 0.25442 0.00000 0.00000 0.00000
5 0.00000 0.00000 0.00000 0.00000 0.25442 0.00000 0.00000
6 0.00000 0.00000 0.00000 0.00000 0.00000 0.25442 0.00000
7 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.25442
Standard Deviations: 0.5044 0.5044 0.5044 0.5044 0.5044 0.5044 0.5044
```

```
getR(fit3o)
```

```
id 1
Conditional variance covariance matrix
      1      2      3      4      5      6      7
1 0.2548 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
2 0.0000 0.2548 0.0000 0.0000 0.0000 0.0000 0.0000
3 0.0000 0.0000 0.2548 0.0000 0.0000 0.0000 0.0000
4 0.0000 0.0000 0.0000 0.2548 0.0000 0.0000 0.0000
5 0.0000 0.0000 0.0000 0.0000 0.2548 0.0000 0.0000
```

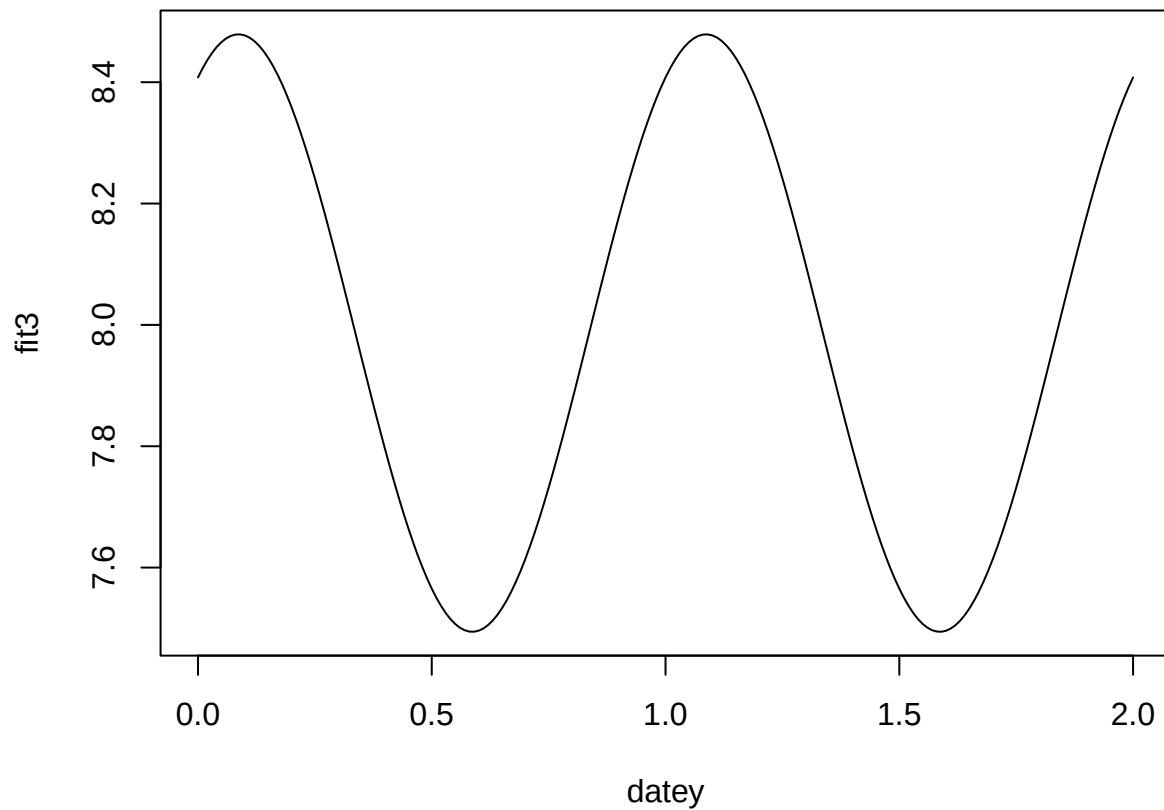
```
6 0.0000 0.0000 0.0000 0.0000 0.0000 0.2548 0.0000
7 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.2548
Standard Deviations: 0.50478 0.50478 0.50478 0.50478 0.50478 0.50478 0.50478
```

Estimating seasonal pattern:

```
preds <- data.frame(datey = seq(0,2,.01))
preds$birthy <- preds$datey - 2      # to move birth out of the way

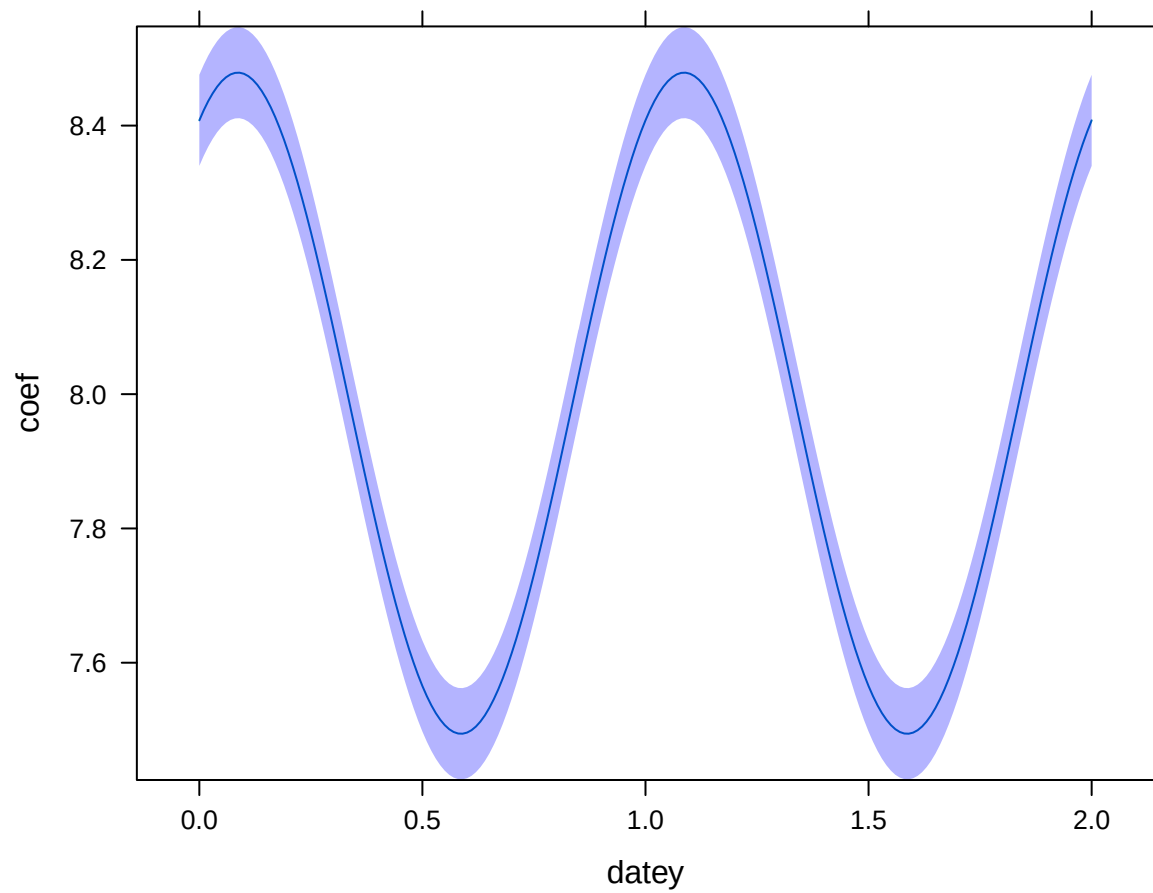
preds$fit3 <- predict(fit3, newdata = preds, level = 0)

with(preds, plot(datey, fit3, type = 'l'))
```



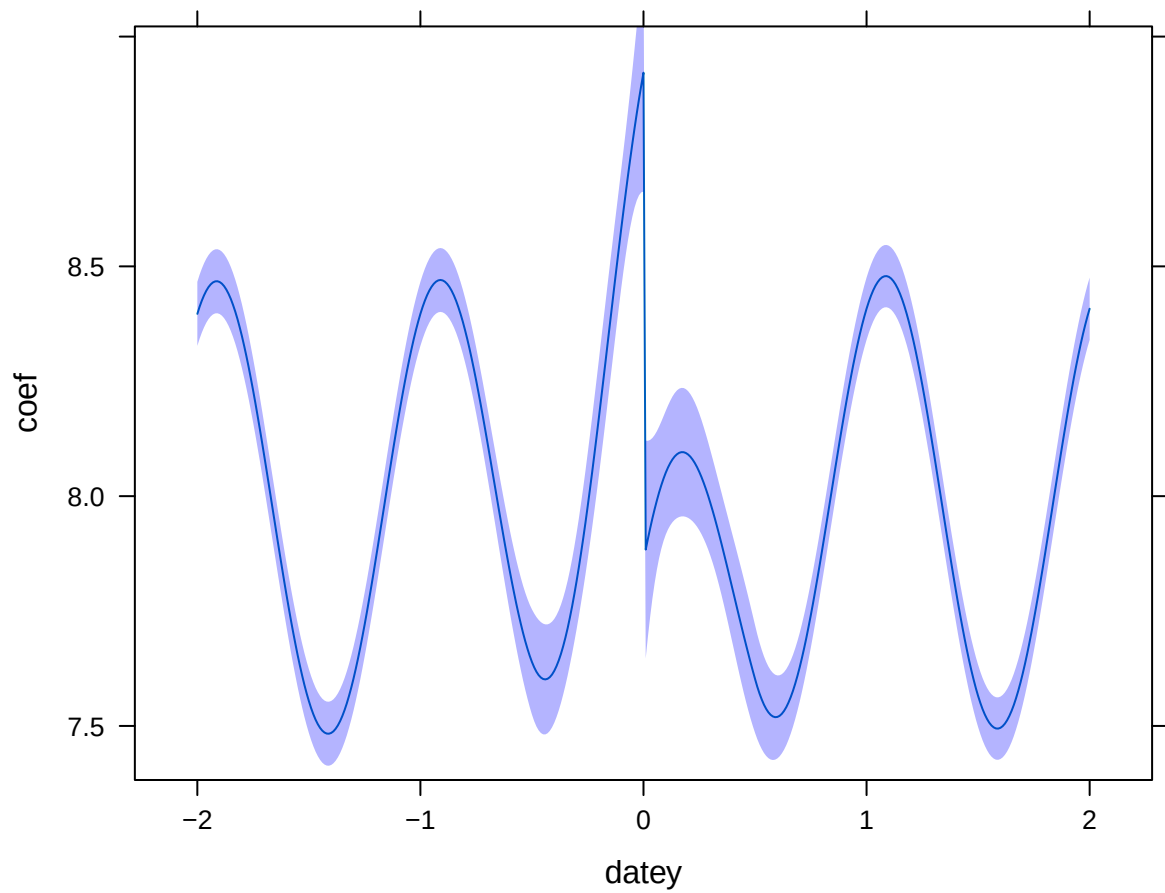
```
ww <- as.data.frame(wald(fit3o, pred = preds))  
  
xyplot(coef ~ datey, ww, type = 'l',  
       lower = ww$L2,
```

```
upper = ww$U2,  
subscripts = TRUE) +  
layer(panel.fit(...))
```



```
ww <- as.data.frame(wald(fit3o, pred = pred))

xyplot(coef ~ datey, ww, type = 'l',
       lower = ww$L2,
       upper = ww$U2,
       subscripts = TRUE) +
  layer(panel.fit(...))
```



Combines seasonal and birth effects showing predicted patterns for a birth on January 1.

To isolate seasonal and birth effects we would need to reparameterize the model to allow unlinking the variable used for calendar date from the variable used for time pre/post birth.

This is left as an exercise. (Challenge: medium)

Solution:

Create a separate variable for days relative to birth

```
head(dd)
```

	id	obs	sleep	birth_date	date	reg_date	birthy	datey
1	1	1	7.088871	5298	1197	288	14.51507	3.279452
1001	1	2	7.336316	5298	1891	288	14.51507	5.180822
2001	1	3	7.909879	5298	2655	288	14.51507	7.273973
3001	1	4	7.489981	5298	3441	288	14.51507	9.427397
4001	1	5	7.087731	5298	4084	288	14.51507	11.189041
5001	1	6	6.920117	5298	4893	288	14.51507	13.405479

```
dd$post_birthy <- with(dd, datey - birthy)
```

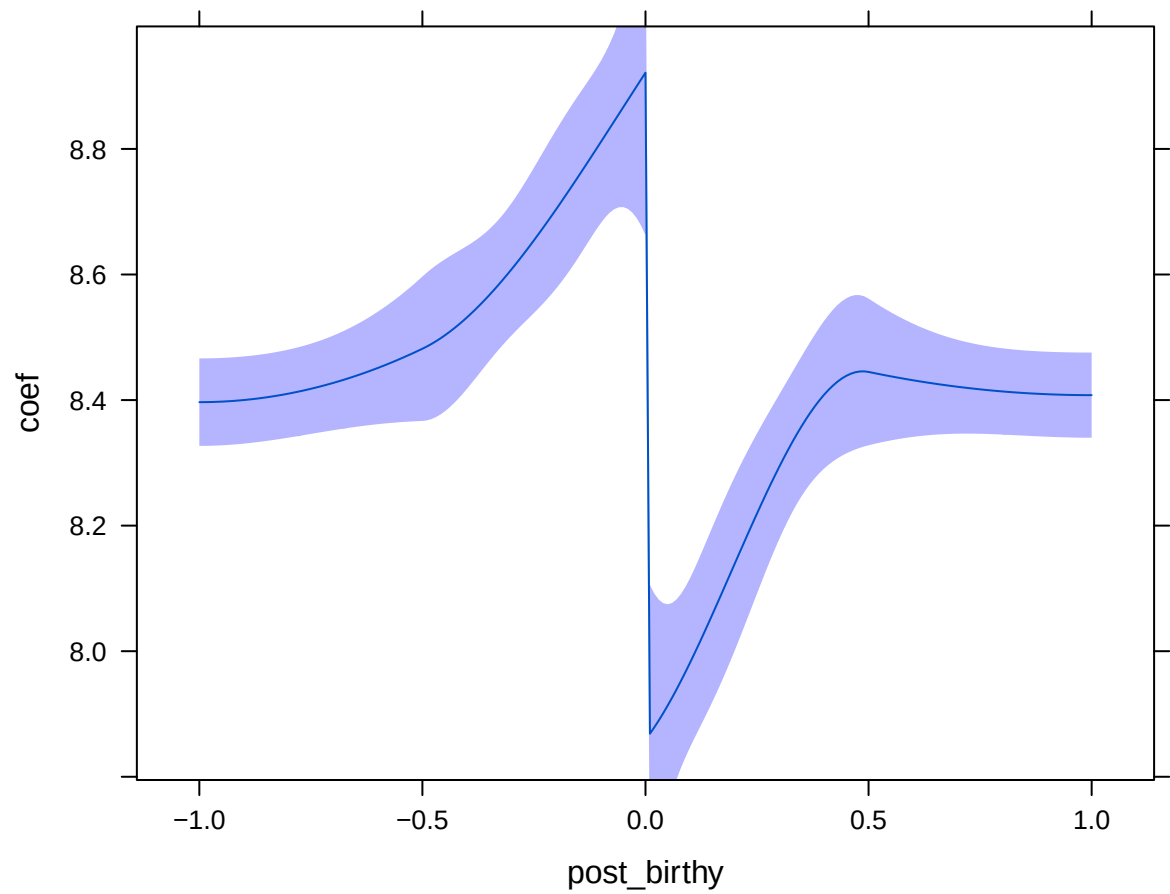
```
fit3b <- lme(sleep ~ sp2(post_birthy) + Sin(2*pi*datey) , dd, random = ~ 1 | id, method = 'ML')
predb <- expand.grid(post_birthy = seq(-1,1,.01), datey = 0)
ww <- as.data.frame( wald(fit3b, pred = predb) )
head(ww)
```

	coef	se	U2	L2	p-value	t-value	DF	post_birthy	datey
1	8.396669	0.03482825	8.466326	8.327013	0	241.0879	5991	-1.00	0

2	8.396703	0.03482662	8.466356	8.327050	0	241.1002	5991	-0.99	0
3	8.396805	0.03482180	8.466449	8.327162	0	241.1365	5991	-0.98	0
4	8.396976	0.03481399	8.466604	8.327348	0	241.1955	5991	-0.97	0
5	8.397214	0.03480350	8.466821	8.327607	0	241.2749	5991	-0.96	0
6	8.397520	0.03479081	8.467102	8.327939	0	241.3718	5991	-0.95	0
	L.(Intercept) L.sp2(post_birthy)D1(0) L.sp2(post_birthy)D2(0)								
1	1.00000000		-0.75000000			0.25000000			
2	1.00000000		-0.74990000			0.24995000			
3	1.00000000		-0.74960000			0.24980000			
4	1.00000000		-0.74910000			0.24955000			
5	1.00000000		-0.74840000			0.24920000			
6	1.00000000		-0.74750000			0.24875000			
	L.sp2(post_birthy)D3(0) L.sp2(post_birthy)C(0).0 L.sp2(post_birthy)C(0).1								
1		-0.05208333		0.00000000		0.00000000			
2		-0.05207083		0.00000000		0.00000000			
3		-0.05203333		0.00000000		0.00000000			
4		-0.05197083		0.00000000		0.00000000			
5		-0.05188333		0.00000000		0.00000000			
6		-0.05177083		0.00000000		0.00000000			
	L.sp2(post_birthy)C(0).2 L.sp2(post_birthy)C(0).3 L.Sin(2 * pi * datey)1								
1		0.00000000		0.00000000		0.00000000			
2		0.00000000		0.00000000		0.00000000			
3		0.00000000		0.00000000		0.00000000			
4		0.00000000		0.00000000		0.00000000			
5		0.00000000		0.00000000		0.00000000			

6	0.00000000	0.00000000	0.00000000
	L.Sin(2 * pi * datey)2		
1	1.00000000		
2	1.00000000		
3	1.00000000		
4	1.00000000		
5	1.00000000		
6	1.00000000		

```
xyplot(coef ~ post_birthy, ww, type = 'l',
       lower = ww$L2,
       upper = ww$U2,
       subscripts = TRUE) +
layer(panel.fit(...))
```



Fitting higher harmonics

```
fit4 <- lme(sleep ~ sp2(datey - birthy) +
            Sin(1 * 2 * pi * datey) +
            Sin(2 * 2 * pi * datey) +
            Sin(3 * 2 * pi * datey)
            , dd, random = ~ 1 | id)
summary(fit4)
```

Linear mixed-effects model fit by REML

Data: dd

	AIC	BIC	logLik
	13690.76	13800.38	-6829.378

Random effects:

Formula: ~1 | id

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

StdDev:	0.9938603	0.5048109
---------	-----------	-----------

Fixed effects: sleep ~ sp2(datey - birthy) + Sin(1 * 2 * pi * datey) + Sin(2 * 2 * pi * datey) + Sin(3 * 2 * pi * datey)

	Value	Std.Error	DF	t-value	p-value
(Intercept)	8.497015	0.12920	5987	65.76594	0.0000
sp2(datey - birthy)D1(0)	1.022949	1.67225	5987	0.61172	0.5407
sp2(datey - birthy)D2(0)	-0.993524	12.04851	5987	-0.08246	0.9343
sp2(datey - birthy)D3(0)	-9.476648	36.37328	5987	-0.26054	0.7945
sp2(datey - birthy)C(0).0	-1.061064	0.17975	5987	-5.90301	0.0000
sp2(datey - birthy)C(0).1	-0.067145	2.43268	5987	-0.02760	0.9780

```

sp2(datey - birthy)C(0).2    8.057741   17.47070 5987   0.46121   0.6447
sp2(datey - birthy)C(0).3 -27.625040  52.62973 5987  -0.52489   0.5997
Sin(1 * 2 * pi * datey)1     0.254924    0.00926 5987  27.54186   0.0000
Sin(1 * 2 * pi * datey)2     0.421517    0.00921 5987  45.78260   0.0000
Sin(2 * 2 * pi * datey)1    -0.010929    0.00932 5987  -1.17309   0.2408
Sin(2 * 2 * pi * datey)2    -0.001708    0.00913 5987  -0.18709   0.8516
Sin(3 * 2 * pi * datey)1    -0.009974    0.00922 5987  -1.08218   0.2792
Sin(3 * 2 * pi * datey)2    -0.007817    0.00918 5987  -0.85177   0.3944

```

Standardized Within-Group Residuals:

Min	Q1	Med	Q3	Max
-3.749540694	-0.631159258	-0.004632081	0.641790295	3.523068562

Number of Observations: 7000

Number of Groups: 1000

We can test higher harmonics with a Wald test or with a LR test

```
wald(fit4, 'Sin\\(3')
```

	numDF	denDF	F-value	p-value
Sin\\(3	2	5987	0.9537905	0.38534

	Estimate	Std.Error	DF	t-value	p-value	Lower	Upper
Sin(3 * 2 * pi * datey)1	-0.009974	0.009216	5987	-1.082177	0.27922	-0.028041	0.95
Sin(3 * 2 * pi * datey)2	-0.007817	0.009177	5987	-0.851774	0.39437	-0.025807	0.95

```
Sin(3 * 2 * pi * datey)1 0.008094
Sin(3 * 2 * pi * datey)2 0.010173
```

```
wald(fit4, 'Sin\\([23]')
```

	numDF	denDF	F-value	p-value				
Sin\\([23]	4	5987	0.8226071	0.51051				
			Estimate	Std.Error	DF	t-value	p-value	Lower 0.95
Sin(2 * 2 * pi * datey)1			-0.010929	0.009316	5987	-1.173091	0.24081	-0.029193
Sin(2 * 2 * pi * datey)2			-0.001708	0.009129	5987	-0.187089	0.85160	-0.019605
Sin(3 * 2 * pi * datey)1			-0.009974	0.009216	5987	-1.082177	0.27922	-0.028041
Sin(3 * 2 * pi * datey)2			-0.007817	0.009177	5987	-0.851774	0.39437	-0.025807
			Upper	0.95				
Sin(2 * 2 * pi * datey)1			0.007335					
Sin(2 * 2 * pi * datey)2			0.016189					
Sin(3 * 2 * pi * datey)1			0.008094					
Sin(3 * 2 * pi * datey)2			0.010173					

```
wald(fit4, 'Sin\\([123]')
```

	numDF	denDF	F-value	p-value				
Sin\\([123]	6	5987	473.4823	<.00001				
			Estimate	Std.Error	DF	t-value	p-value	Lower 0.95
Sin(1 * 2 * pi * datey)1			0.254924	0.009256	5987	27.541861	<.00001	0.236779
Sin(1 * 2 * pi * datey)2			0.421517	0.009207	5987	45.782595	<.00001	0.403468
Sin(2 * 2 * pi * datey)1			-0.010929	0.009316	5987	-1.173091	0.24081	-0.029193
Sin(2 * 2 * pi * datey)2			-0.001708	0.009129	5987	-0.187089	0.85160	-0.019605

```

Sin(3 * 2 * pi * datey)1 -0.009974 0.009216 5987 -1.082177 0.27922 -0.028041
Sin(3 * 2 * pi * datey)2 -0.007817 0.009177 5987 -0.851774 0.39437 -0.025807
      Upper 0.95
Sin(1 * 2 * pi * datey)1 0.273069
Sin(1 * 2 * pi * datey)2 0.439566
Sin(2 * 2 * pi * datey)1 0.007335
Sin(2 * 2 * pi * datey)2 0.016189
Sin(3 * 2 * pi * datey)1 0.008094
Sin(3 * 2 * pi * datey)2 0.010173

```

```
anova(update(fit3, method = 'ML'), update(fit4, method = "ML"))
```

	Model	df	AIC	BIC	logLik	Test	L.Ratio
update(fit3, method = "ML")	1	12	13645.35	13727.59	-6810.676		
update(fit4, method = "ML")	2	16	13650.06	13759.71	-6809.027	1 vs 2	3.29655
						p-value	
update(fit3, method = "ML")							
update(fit4, method = "ML")						0.5095	

Note how close the corresponding p-values are from the two tests

Hand-rolled parametric splines

Many parametric splines are easily make ‘by hand’.

As long as a spline adds polynomial degrees monotonically, either from the left to the right, or from the right to the

left, it can be easily specified with ‘plus’ functions:

```
plus <- function(x) x * (x > 0)
```

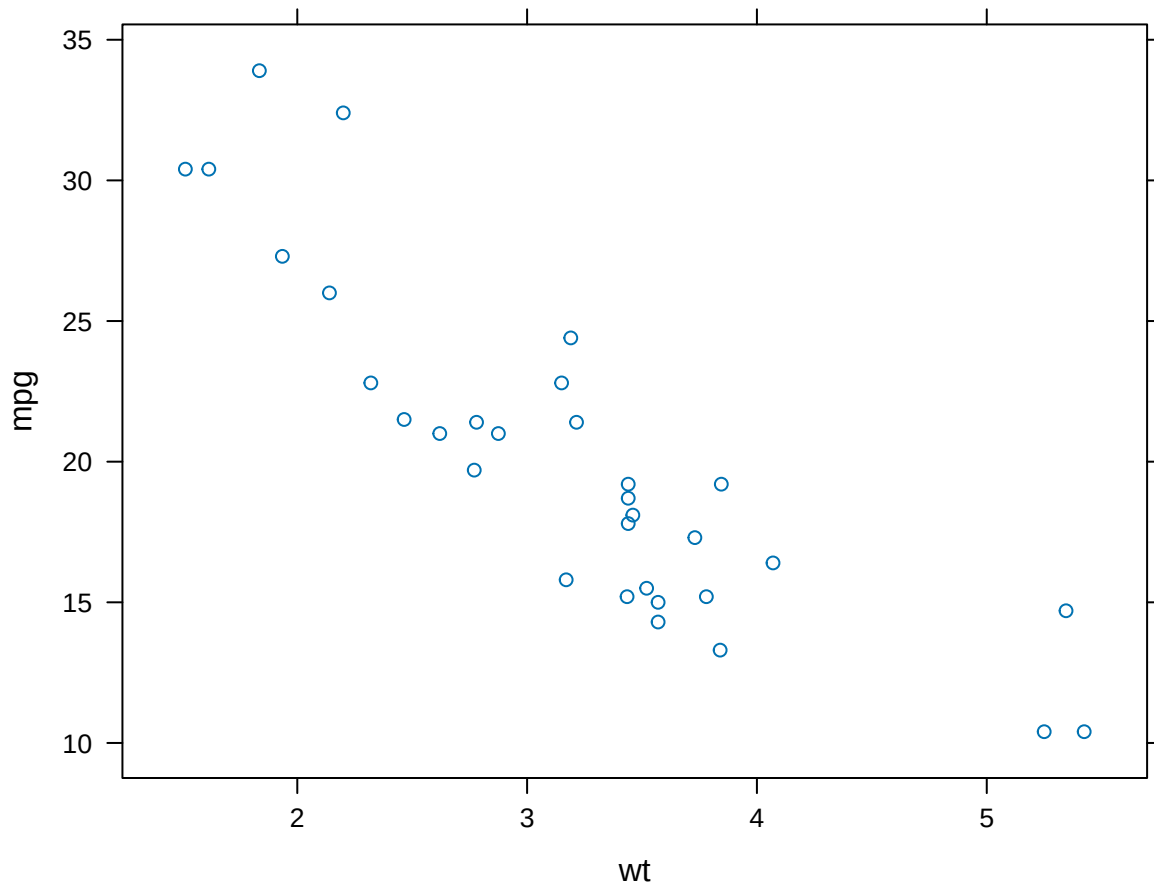
For example, a piece-wise linear function with knots at 0 and at 1 could be specified as:

```
y ~ x + plus(x) + plus(x - 1)
```

```
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

```
xyplot(mpg ~ wt, mtcars)
```



```
plus <- function(x) x * (x>0)
fit <- lm(mpg ~ disp + wt + plus(wt - 3) + plus(wt - 4), mtcars)
```

```
summary(fit)
```

Call:

```
lm(formula = mpg ~ disp + wt + plus(wt - 3) + plus(wt - 4), data = mtcars)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-3.0400	-1.6780	-0.7629	0.8295	5.3393

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	42.97993	3.25485	13.205	2.7e-13	***
disp	-0.01890	0.00834	-2.266	0.031701	*
wt	-6.56004	1.46556	-4.476	0.000124	***
plus(wt - 3)	5.05495	3.04939	1.658	0.108955	
plus(wt - 4)	0.52160	3.12072	0.167	0.868506	

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

Residual standard error: 2.525 on 27 degrees of freedom

Multiple R-squared: 0.8471, Adjusted R-squared: 0.8245

F-statistic: 37.4 on 4 and 27 DF, p-value: 1.213e-10

```
wald(fit, 'plus') # individual components not sig. but overall sig.
```

	numDF	denDF	F-value	p-value
plus	2	27	5.843264	0.00779

	Estimate	Std.Error	DF	t-value	p-value	Lower 0.95	Upper 0.95
plus(wt - 3)	5.054949	3.049385	27	1.657695	0.10895	-1.201872	11.311771
plus(wt - 4)	0.521595	3.120718	27	0.167139	0.86851	-5.881589	6.924778

Exercise: interpret the coefficients of this model

```
L <- rbind(
  "Slope wt<3" = c(0,0,1,0,0),
  "Slope 3<wt<4" = c(0,0,1,1,0),
  "Slope 4<wt" = c(0,0,1,1,1),
  "Change at 3" = c(0,0,0,1,0),
  "Change at 4" = c(0,0,0,0,1)
)
```

```
wald(fit, L)
```

	numDF	denDF	F-value	p-value
1	3	27	7.579704	0.00078

	Estimate	Std.Error	DF	t-value	p-value	Lower 0.95	Upper 0.95
Slope wt<3	-6.560040	1.465563	27	-4.476122	0.00012	-9.567127	-3.552953
Slope 3<wt<4	-1.505090	2.449412	27	-0.614470	0.54405	-6.530868	3.520687
Slope 4<wt	-0.983496	1.549031	27	-0.634910	0.53083	-4.161845	2.194854

Change at 3	5.054949	3.049385	27	1.657695	0.10895	-1.201872	11.311771
Change at 4	0.521595	3.120718	27	0.167139	0.86851	-5.881589	6.924778

The previous spline is equivalent to the following 'gsp' spline:

```
sp1 <- function(x) gsp(x, c(3,4), c(1,1,1), c(0,0))

fits <- lm(mpg ~ disp + sp1(wt), mtcars)
summary(fits)
```

Call:

```
lm(formula = mpg ~ disp + sp1(wt), data = mtcars)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-3.0400	-1.6780	-0.7629	0.8295	5.3393

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	42.97993	3.25485	13.205	2.7e-13 ***
disp	-0.01890	0.00834	-2.266	0.031701 *
sp1(wt)D1(0)	-6.56004	1.46556	-4.476	0.000124 ***
sp1(wt)C(3).1	5.05495	3.04939	1.658	0.108955
sp1(wt)C(4).1	0.52160	3.12072	0.167	0.868506

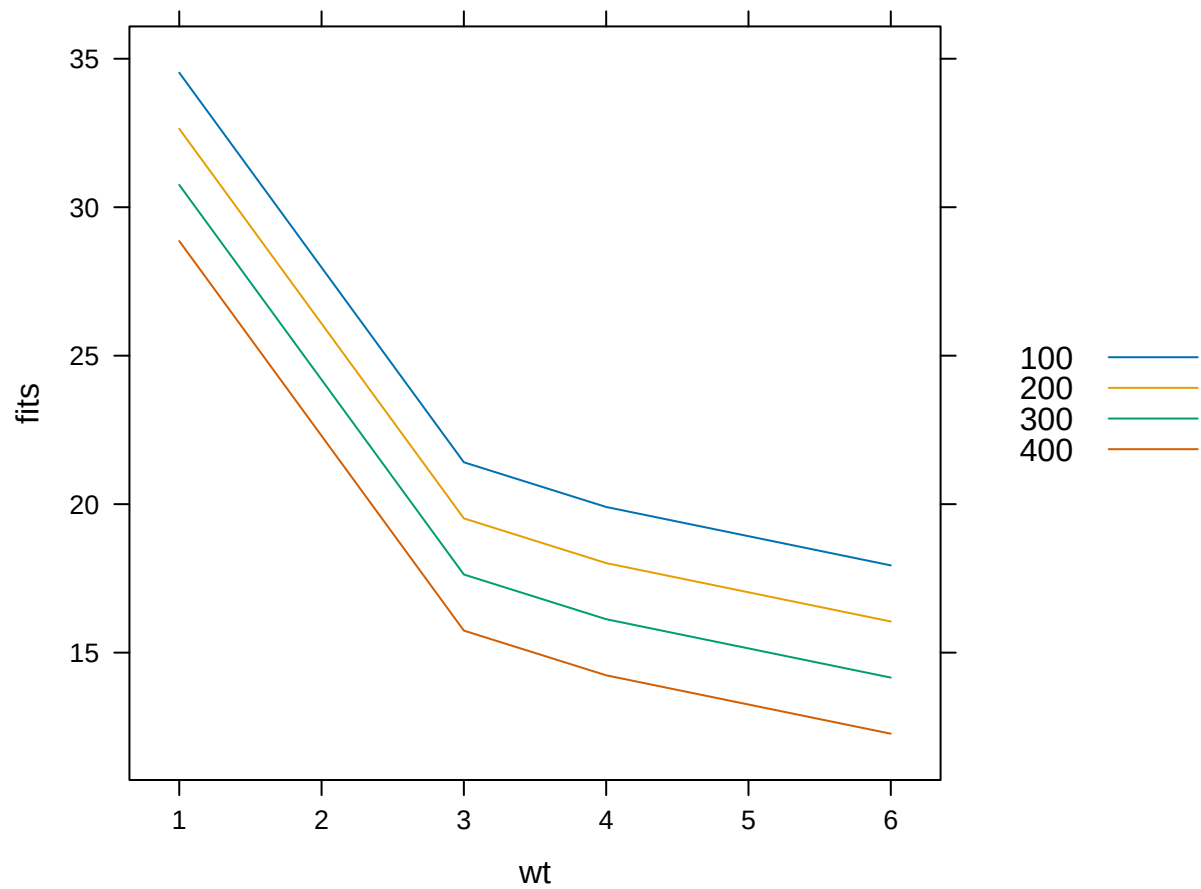
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.525 on 27 degrees of freedom

Multiple R-squared: 0.8471, Adjusted R-squared: 0.8245

F-statistic: 37.4 on 4 and 27 DF, p-value: 1.213e-10

```
pred <- with(mtcars, pred.grid(wt= seq(1,6,.01), disp = seq(100,400,100)))
pred$fits <- predict(fits, newdata = pred)      # no levels argument since this is 'monolevel'
xyplot(fits ~ wt, pred, groups = disp, type = 'l', auto.key = T)
```



```
car::compareCoefs(fit, fits)
```

Calls:

```
1: lm(formula = mpg ~ disp + wt + plus(wt - 3) + plus(wt - 4), data =  
  mtcars)
```

```
2: lm(formula = mpg ~ disp + sp1(wt), data = mtcars)
```

	Model 1	Model 2
(Intercept)	42.98	42.98
SE	3.25	3.25
disp	-0.01890	-0.01890
SE	0.00834	0.00834
wt	-6.56	
SE	1.47	
plus(wt - 3)	5.05	
SE	3.05	
plus(wt - 4)	0.522	
SE	3.121	
sp1(wt)D1(0)		-6.56
SE		1.47
sp1(wt)C(3).1		5.05

SE	3.05
sp1(wt)C(4).1	0.522
SE	3.121

Fitting a different spline

```
sp2 <- function(x) gsp(x, 4, c(2,1), 1)
fitq <- lm(mpg ~ disp + sp2(wt), mtcars)
summary(fitq)
```

Call:

```
lm(formula = mpg ~ disp + sp2(wt), data = mtcars)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.8311	-1.5000	-0.9477	0.9182	5.6663

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	50.495411	4.738387	10.657	2.32e-11	***
disp	-0.018532	0.007768	-2.386	0.024050	*
sp2(wt)D1(0)	-13.677911	3.067820	-4.459	0.000122	***
sp2(wt)D2(0)	3.218240	0.905567	3.554	0.001370	**

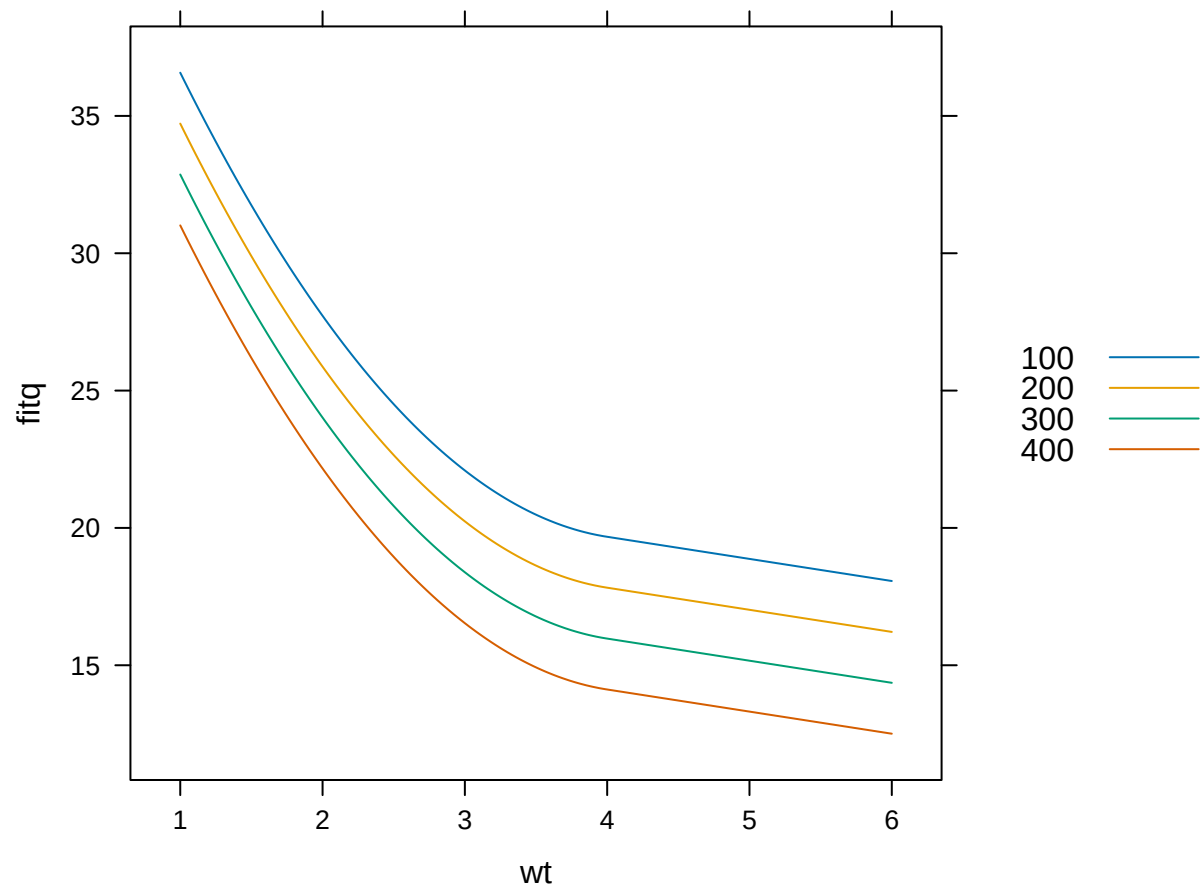
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.464 on 28 degrees of freedom

Multiple R-squared: 0.849, Adjusted R-squared: 0.8329

F-statistic: 52.49 on 3 and 28 DF, p-value: 1.284e-11

```
pred$fitq <- predict(fitq, newdata = pred)
xyplot(fitq ~ wt, pred, groups = disp, type = 'l', auto.key = T)
```

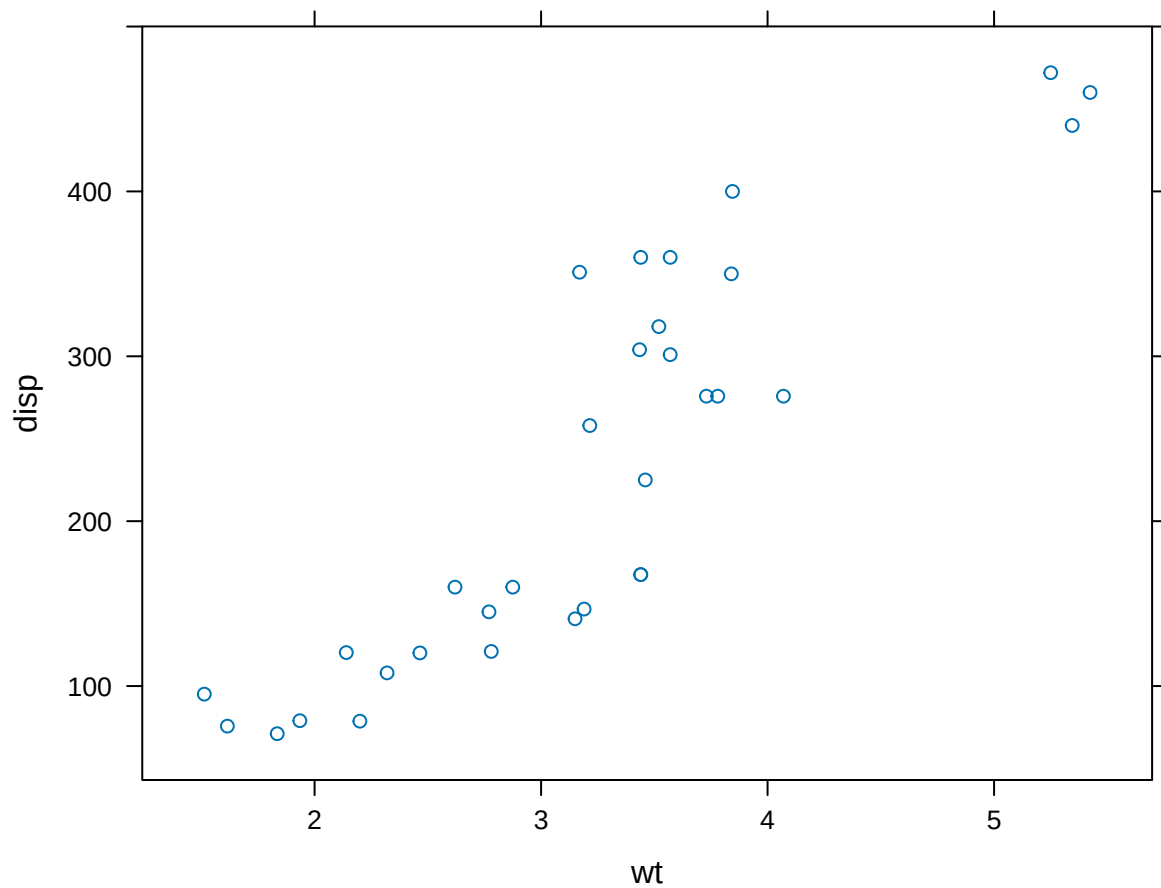


`AIC(fits, fitq)`

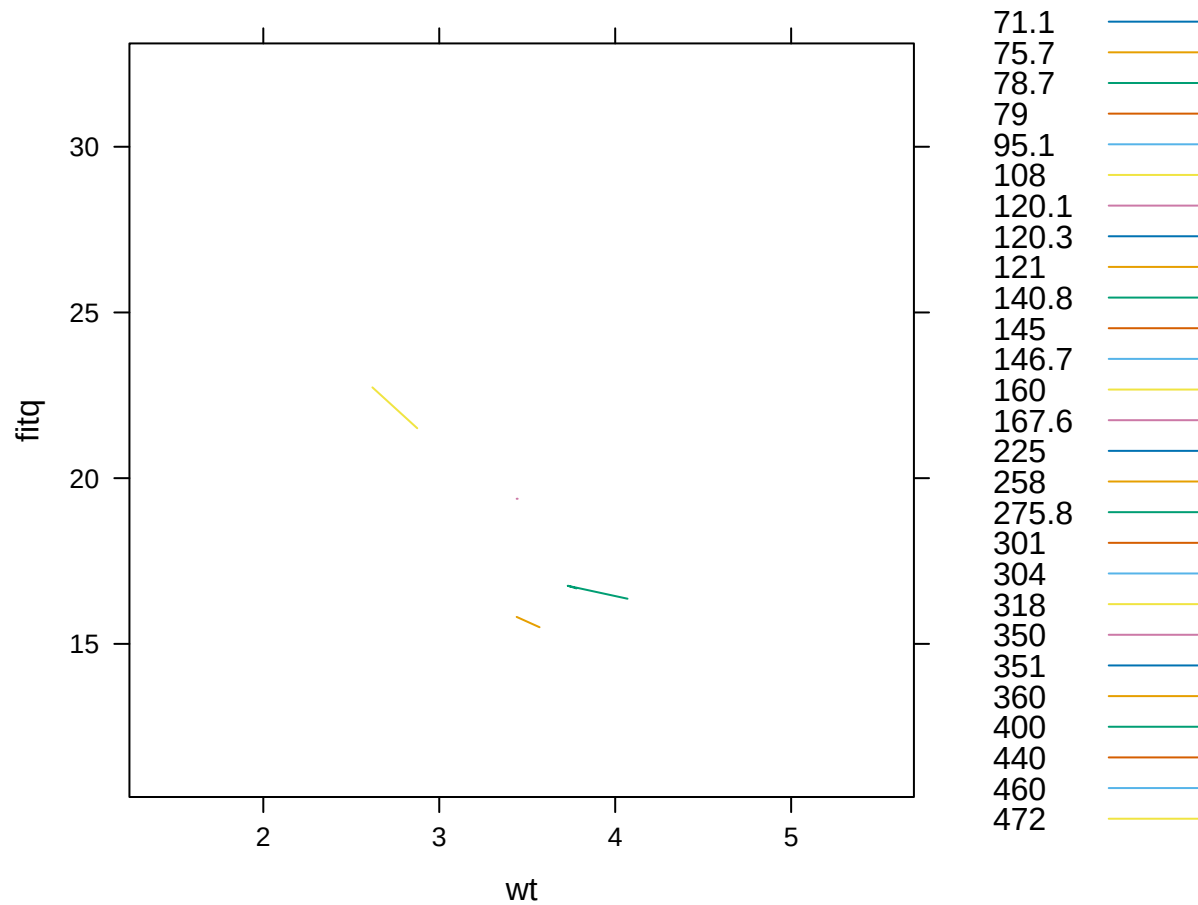
	df	AIC
fits	6	156.6588
fitq	5	154.2543

For a more realistic plot that reflects the relationship among predictors, use the original data but might fill in values for continuity

```
mtcars$fitq <- predict(fitq)
xyplot(displ ~ wt, mtcars)
```



```
xyplot(fitq ~ wt, mtcars, groups = disp, type = 'l', auto.key = T)
```



```
library(p3d)
```

```
Loading required package: rgl
```

```
Attaching package: 'p3d'
```

```
The following objects are masked from 'package:spida2':
```

```
cell, center, ConjComp, dell, disp, ell, ell.conj, ellbox, ellplus,  
ellpt, ellptc, ellpts, ellptsc, elltan, elltanc, na.include, uv
```

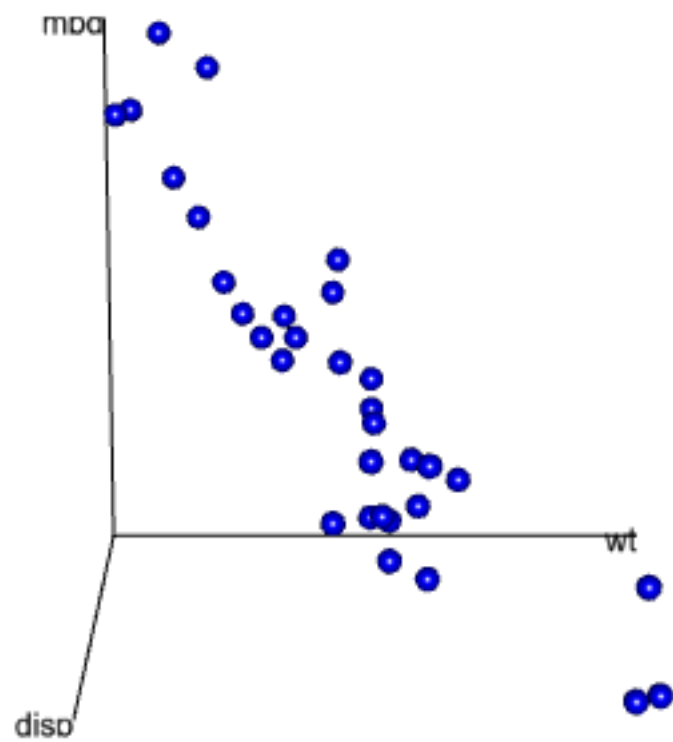
```
Init3d()
```

```
Plot3d(mpg ~ wt + disp, mtcars)
```

```
Use left mouse to rotate, middle mouse (or scroll) to zoom, right mouse to change perspective
```

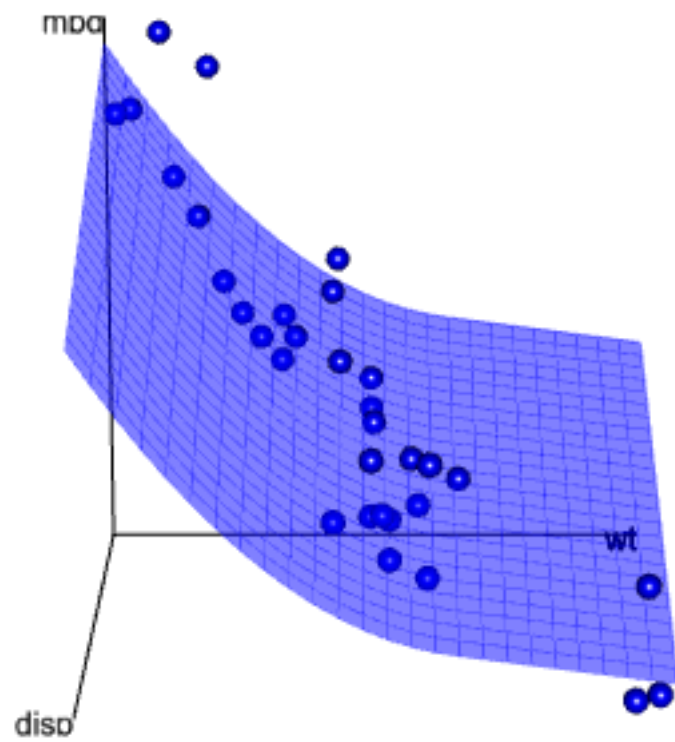
```
rglwidget()
```

```
file:///tmp/RtmpFX9E5Z/file3825a5369f736.html screenshot completed
```



```
Fit3d(fitq)  
rglwidget()
```

file:///tmp/RtmpFX9E5Z/file3825a131400d1.html screenshot completed



```
Fit3d(fits, col = 'pink')  
rglwidget()
```

file:///tmp/RtmpFX9E5Z/file3825a28b36520.html screenshot completed

