

Using splines in regression

Author: Nicholas G Reich, Jeff Goldsmith

*This material is part of the **statsTeachR** project*

*Made available under the Creative Commons Attribution-ShareAlike 3.0 Unported
License: http://creativecommons.org/licenses/by-sa/3.0/deed.en_US*

Today's Lecture

- Spline models
- Penalized spline regression

More info:

- Harrel, *Regression Modeling Strategies*, Chapter 2, PDF handout
- *ISL* Chapter 7

Piecewise linear models

A piecewise linear model (also called a change point model or broken stick model) contains a few linear components

- Outcome is linear over full domain, but with a different slope at different points
- Points where relationship changes are referred to as “change points” or “knots”
- Often there's one (or a few) potential change points

Piecewise linear models

Suppose we want to estimate $E(y|x) = f(x)$ using a piecewise linear model.

- For one knot we can write this as

$$E(y|x) = \beta_0 + \beta_1 x + \beta_2 (x - \kappa)_+$$

where κ is the location of the change point and

$$(x - \kappa)_+ =$$

Interpretation of regression coefficients

$$\mathbb{E}(y|x) = \beta_0 + \beta_1 x + \beta_2 (x - \kappa)_+$$

- $\beta_0 = \mathbb{E}[y|x = 0]$ (assuming $\kappa > 0$)

- $\beta_1 =$

- $\beta_2 =$

- $\beta_1 + \beta_2 =$

Interpretation of regression coefficients

$$\mathbb{E}(y|x) = \beta_0 + \beta_1 x + \beta_2 (x - \kappa)_+$$

- $\beta_0 = \mathbb{E}[y|x = 0]$ (assuming $\kappa > 0$)
- $\beta_1 =$ Expected change in y for a 1-unit increase in x , when $x < \kappa$
- $\beta_2 =$
- $\beta_1 + \beta_2 =$

Interpretation of regression coefficients

$$\mathbb{E}(y|x) = \beta_0 + \beta_1 x + \beta_2 (x - \kappa)_+$$

- $\beta_0 = \mathbb{E}[y|x = 0]$ (assuming $\kappa > 0$)
- $\beta_1 =$ Expected change in y for a 1-unit increase in x , when $x < \kappa$
- $\beta_2 =$ Change in slope between $x < \kappa$ and $x > \kappa$
- $\beta_1 + \beta_2 =$

Interpretation of regression coefficients

$$E(y|x) = \beta_0 + \beta_1 x + \beta_2 (x - \kappa)_+$$

- $\beta_0 = \mathbb{E}[y|x = 0]$ (assuming $\kappa > 0$)
- $\beta_1 =$ Expected change in y for a 1-unit increase in x , when $x < \kappa$
- $\beta_2 =$ Change in slope between $x < \kappa$ and $x > \kappa$
- $\beta_1 + \beta_2 =$ Expected change in y for a 1-unit increase in x , when $x \geq \kappa$

Estimation

- Piecewise linear models are low-dimensional (no need for penalization)
- Parameters are estimated via OLS
- The design matrix is ...

Multiple knots

Suppose we want to estimate $E(y|x) = f(x)$ using a piecewise linear model.

- For multiple knots we can write this as

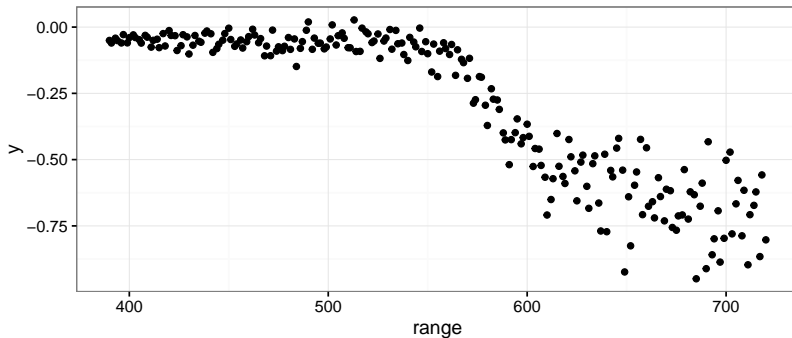
$$E(y|x) = \beta_0 + \beta_1 x + \sum_{k=1}^K \beta_{k+1} (x - \kappa_k)_+$$

where $\{\kappa_k\}_{k=1}^K$ are the locations of the change points

- Note that knot locations are defined before estimating regression coefficients
- Also, regression coefficients are interpreted conditional on the knots.

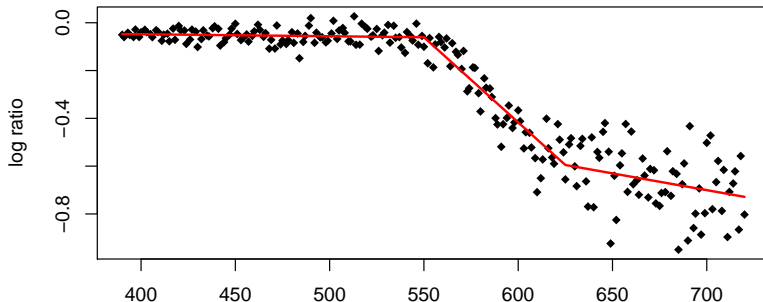
Example: lidar data

```
library(MASS)
library(SemiPar)
data(lidar)
y = lidar$logratio
range = lidar$range
qplot(range, y)
```



Example: lidar data

```
knots <- c(550, 625)
mkSpline <- function(k, x) (x - k > 0) * (x - k)
X.des = cbind(1, range, sapply(knots, FUN=mkSpline, x=range))
colnames(X.des) <- c("intercept", "range", "range1", "range2")
lm.lin = lm(y ~ X.des - 1)
plot(range, y, xlab = "Range", ylab = "log ratio", pch = 18)
points(range, lm.lin$fitted.values, type = 'l', col = "red", lwd = 2)
```



Example: lidar data

```
summary(lm.lin)$coef
```

##		Estimate	Std. Error	t value	Pr(> t)
##	X.desintercept	-1.444288e-02	0.0687353855	-0.2101230	8.337689e-01
##	X.desrange	-8.407376e-05	0.0001426647	-0.5893102	5.562663e-01
##	X.desrange1	-7.042794e-03	0.0003834218	-18.3682689	4.379404e-46
##	X.desrange2	5.723186e-03	0.0005153479	11.1054811	5.554824e-23

Piecewise quadratic and cubic models

Suppose we want to estimate $E(y|x) = f(x)$ using a piecewise quadratic model.

- For multiple knots we can write this as

$$E(y|x) = \beta_0 + \beta_1 x + \beta_1 x^2 + \sum_{k=1}^K \beta_{k+2} (x - \kappa_k)_+^2$$

where $\{\kappa_k\}_{k=1}^K$ are the locations of the change points

- Similar extension for cubics
- Piecewise quadratic models are smooth and have continuous first derivatives

Pros and cons of piecewise models

Piecewise (linear, quadratic, etc) models have several advantages

- Easy construction of basis functions
- Flexible, and don't rely on determining an appropriate form for $f(x)$ using standard functions
- Allow for significance testing on change point slopes
- Fairly direct interpretations

Disadvantages

- knot specification is often arbitrary

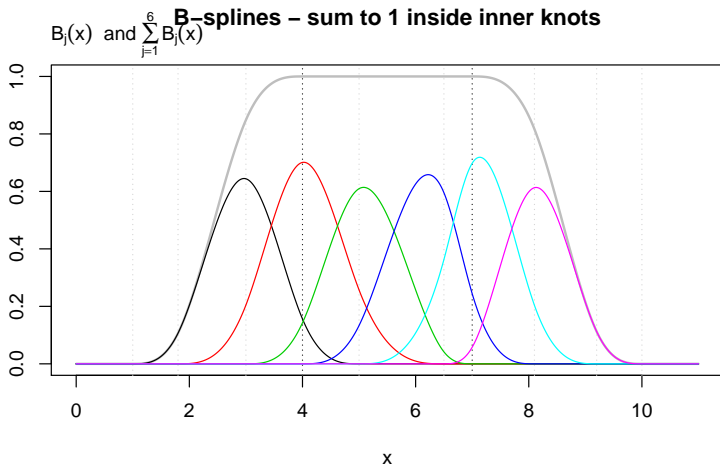
B-splines and natural splines

Characteristics

- Both B-splines and natural splines similarly define a basis over the domain of x
- Can be constrained to have seasonal patterns
- They are made up of piecewise polynomials of a given degree, and have defined derivatives similarly to the piecewise defined functions
- Big advantage over linear splines: parameter estimation is often fairly robust to your choice of knots
- Big disadvantage over linear splines: harder to interpret specific coefficients

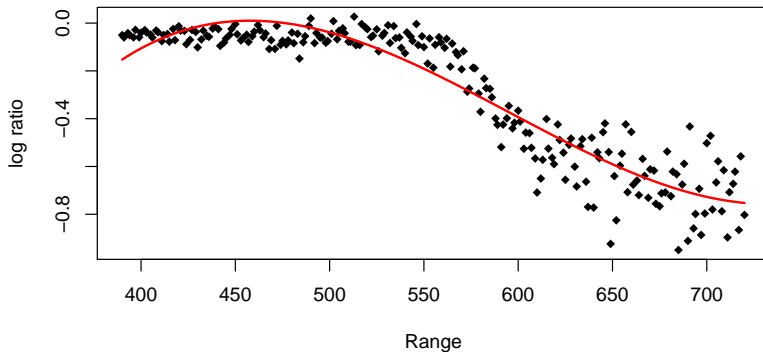
B-splines basis functions

$$E(y|x) = \beta_0 + \sum_{j=1}^6 \beta_j B_j(x)$$



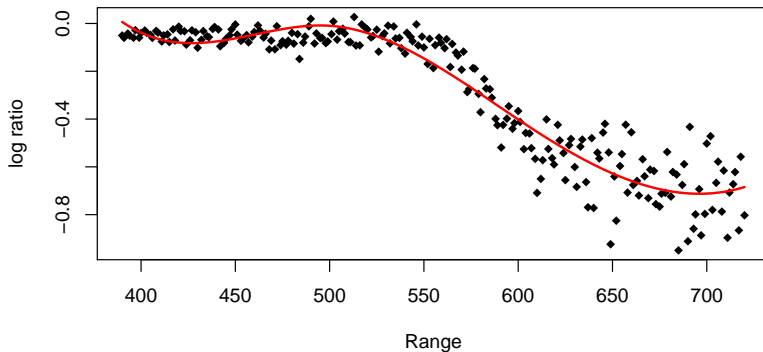
Example: lidar data

```
require(splines)
lm.bs3 = lm(y ~ bs(range, df=3))
plot(range, y, xlab = "Range", ylab = "log ratio", pch = 18)
points(range, lm.bs3$fitted.values, type = 'l', col = "red", lwd = 2)
```



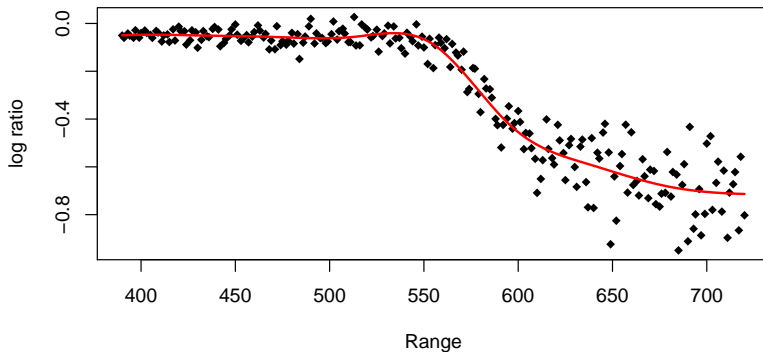
Example: lidar data

```
lm.bs5 = lm(y ~ bs(range, df=5))  
plot(range, y, xlab = "Range", ylab = "log ratio", pch = 18)  
points(range, lm.bs5$fitted.values, type = 'l', col = "red", lwd = 2)
```



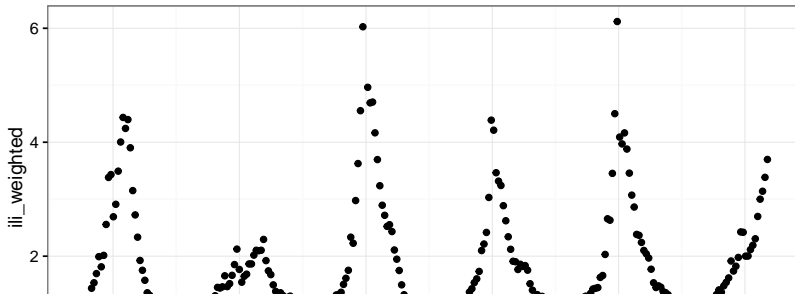
Example: lidar data

```
lm.bs5 = lm(y ~ bs(range, df=10))  
plot(range, y, xlab = "Range", ylab = "log ratio", pch = 18)  
points(range, lm.bs5$fitted.values, type = 'l', col = "red", lwd = 2)
```



Another setting where splines could be useful

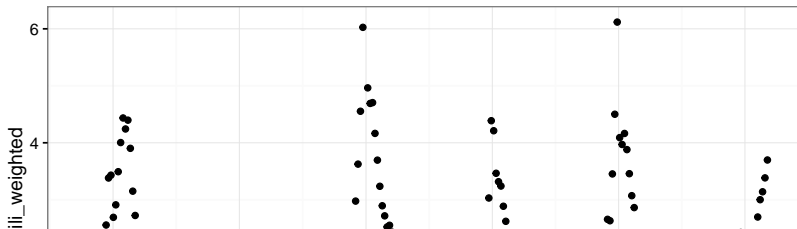
```
library(cdcfluview)
library(dplyr)
library(lubridate)
usflu <- get_flu_data("national", "ilinet", years=2010:2015)
usflu <- mutate(usflu,
                 date = as.Date(paste0(YEAR, sprintf("%02d", WEEK), "00"),
                                format="%Y%W%w"),
                 ili_weighted = X.UNWEIGHTED.ILI,
                 dec_date = decimal_date(date),
                 time_in_year = dec_date%%1)
ggplot(usflu, aes(x=date, y=ili_weighted)) + geom_point()
```



Time series analysis

These observations are time-series data

- Time is a “special” covariate: order matters, independence assumption often violated.
- There are more formal and systematic methods of analyzing time-series data that we are not going to discuss today, e.g. ARIMA models. (If interested, consider STAT 597TS.)
- A few nice resources: Hyndman and Athanasopoulos’s *Forecasting: Principles and Practice*, at <https://www.otexts.org/fpp>, or Shumway and Stoffer, *Time Series Analysis and Its Applications: with R examples*.

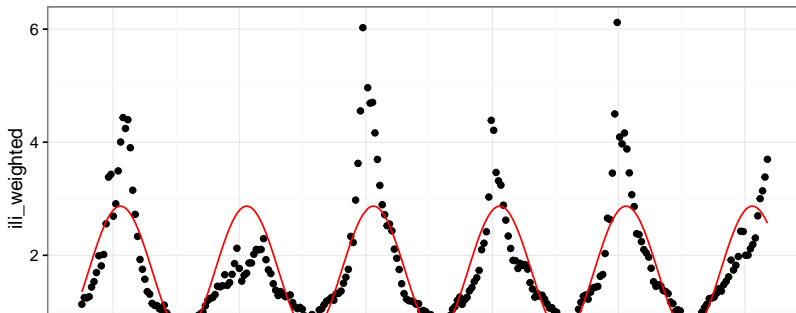


Fit a trigonometric model

We could fit a seasonal model like this:

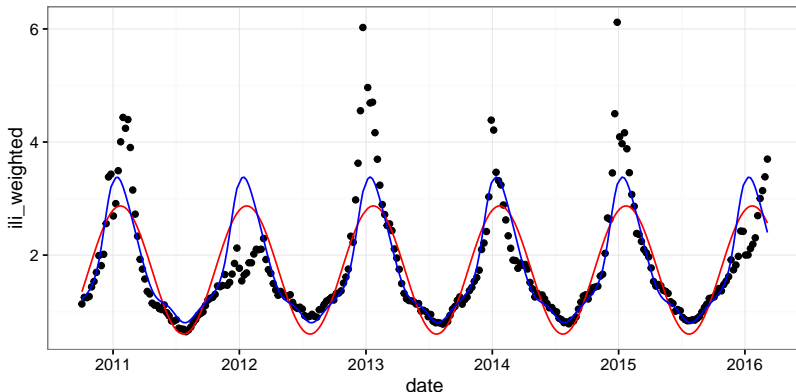
$$\mathbb{E}[y_t] = \beta_0 + \beta_1 \cdot \sin(2\pi t) + \cos(1\pi t)$$

```
usflu <- usflu[!is.na(usflu$dec_date),] ## sloppy: removing one "week 5"  
trig_mdl <- lm(ili_weighted ~ sin(dec_date*2*pi) + cos(dec_date*2*pi),  
              data=usflu)  
usflu$trig_preds <- predict(trig_mdl)  
ggplot(usflu, aes(x=date, y=ili_weighted)) + geom_point() +  
  geom_line(aes(y=trig_preds), color="red")
```



Fit a smoothing splines model

```
library(mgcv)
spline_md1 <- gam(ili_weighted ~ s(time_in_year, bs="cc"), data=usflu)
usflu$spl_preds <- predict(spline_md1)
ggplot(usflu, aes(x=date, y=ili_weighted)) + geom_point() +
  geom_line(aes(y=trig_preds), color="red") +
  geom_line(aes(y=spl_preds), color="blue")
```



Take-home points for spline approaches (1)

Splines can flexibly model non-linear relationships

- Can improve model fit because of relaxed linearity assumptions.
- Caveat: spline models require careful graphical interpretation, slopes may not be easily available/interpretable

`figs/pediatrics-example.png`

Take-home points for spline approaches (2)

Do you want control over your knots?

- Your application may have explicit “change-points” (i.e. interrupted time-series)
- In most cases, you do not want your spline model to be sensitive to user input (i.e. knot placement)
- “Penalized splines” can reduce this sensitivity at the cost of more complex model and estimation (More in *ISL* Chapter 7, Biostat Methods 3, anything about Generalized Additive Models (e.g. `mgcv` package and `gam()` function), one of your projects?).