STA2201H Methods of Applied Statistics II

Monica Alexander

Week 11: Splines

Notes

Presentations next week

- Let me know whether you need to do it via zoom
- Send me your slides by 830am next Wednesday so I can collate the list
    - pdf preferred
- Aim for 5 mins, brief overview
- I'll bring coffee for in person

Reading

- Eilers, P.H. and Marx, B.D., 1996. Flexible smoothing with B-splines and penalties. Statistical science, pp.89-102.
- Friedman, J., Hastie, T. and Tibshirani, R., 2001. The elements of statistical learning. Chapter 5
- James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. An introduction to statistical learning. Chapter 7
- BDA Chapter 20

The end goal for today

What are we doing: Bayesian Penalized B-Splines (P-Splines) Regression

How do we get there

1. What are splines?
2. What are B-splines?
3. How to fit B-Splines regression?
4. How are they penalized?
5. How to fit P-Splines in a Bayesian framework?

Moving beyond linearity

- Replace the vector of inputs X with transformations of X
- Use linear models in the new space of derived inputs

i.e. instead of fitting a linear model in X we fit the model

$$y_i = \alpha_0 + \alpha_1 b_1(x_i) + \alpha_2 b_2(x_i) + \cdots + +\alpha_K b_K(x_i) + \epsilon_i$$

- the functions $b_k(.)$ are called **basis functions** and are fixed and known i.e. we choose them.
- can think of this as a standard linear model so all the usual techniques apply (least squares, standard errors on coefficients, etc)

Basis functions

We have

$$y_i = \mu\left(x_i\right) + \varepsilon_i$$
$$\varepsilon_i \sim N\left(0, \sigma_y^2\right)$$

with $\mu(x_i) = \sum_{k=1}^{K} b_k(x_i)\alpha_k$.

What to choose for $b(.)$?

- simplest: piece-wise constant
- piece wise polynomials
- ... regression splines

Piecewise polynomials

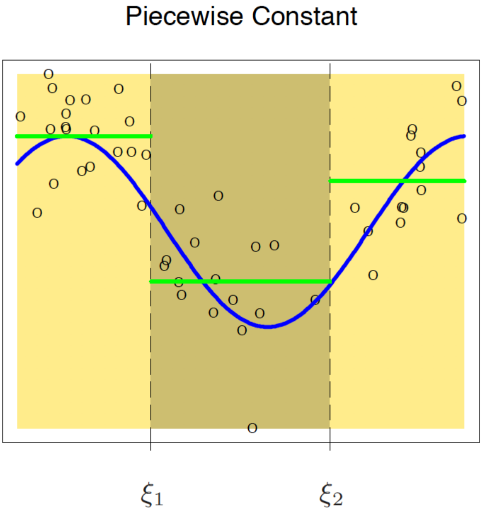
- Divide the domain of X into K intervals defined by **knot points** $\tau_1, \tau_2, \ldots \tau_{K+1}$, a represent y as a separate polynomial in each interval

E.g. piece wise constant with two knot points

$b_1(X) = I\,(X < \tau_1) \quad b_2(X) = I\,(\tau_1 \leq X < \tau_2)$
$b_3(X) = I\,(\tau_2 \leq X)$

HTF fig 5.1

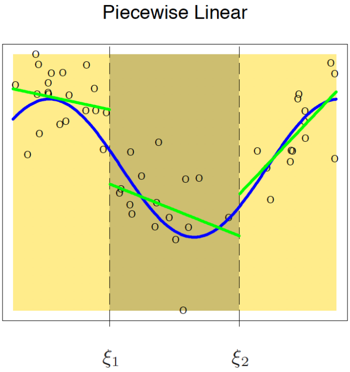
(Knot points are denoted with ξ_k)



Piecewise polynomials

Slightly more complicated: piecewise linear. Three additional basis functions are needed:

$$b_{m+3} = b_m(X)X, m = 1, \ldots, 3$$



(HTF Fig 5.1)

Piecewise polynomials with constraints

We would usually want to constrain the pieces to be continuous at the knot points

e.g. in the piecewise linear case

$$b_1(X) = 1, \quad b_2(X) = X, \quad b_3(X) = (X - \tau_1)_+, \quad b_4(X) = (X - \tau_2)_+$$

where $(X - \tau_1)_+ = \begin{cases} (X - \tau_1) & \text{if } x > \tau_1 \\ 0 & \text{otherwise} \end{cases}$

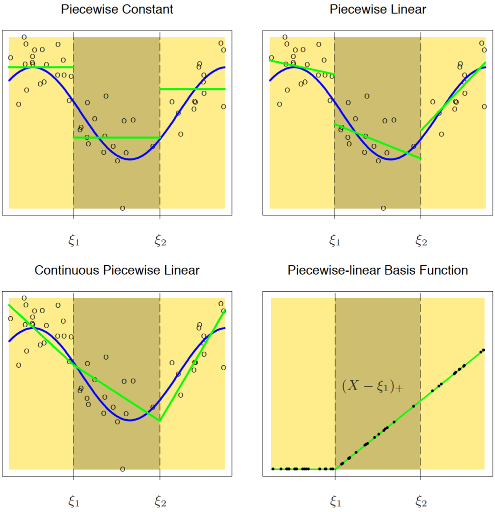
Remember that these add so

$$y_i = \sum_K \alpha_k b_k(x_i) + \epsilon_i$$

So the model is linear with changing slope.

HTF fig 5.1

Bottom right hand panel is the function $(X - \tau_1)_+$



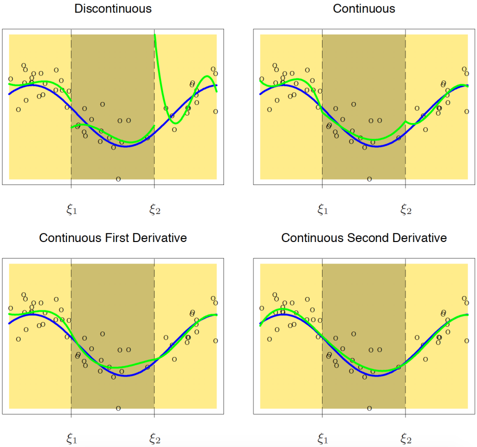
Regression splines

- Piece-wise polynomials with continuity restrictions are called splines.
- In particular, a degree-d spline is a piecewise degree-d polynomial with continuity in derivatives up to degree $d-1$ at each knot.
- We often prefer smoother functions; these can be achieved by increasing the order of the local polynomial

Commonly chosen are **cubic splines**: third-order polynomial with continuity in first and second derivatives

Cubic polynomials

HTF, Figure 5.2. Bottom RH panel is a cubic spline



Cubic spline basis

For the case with two knots,

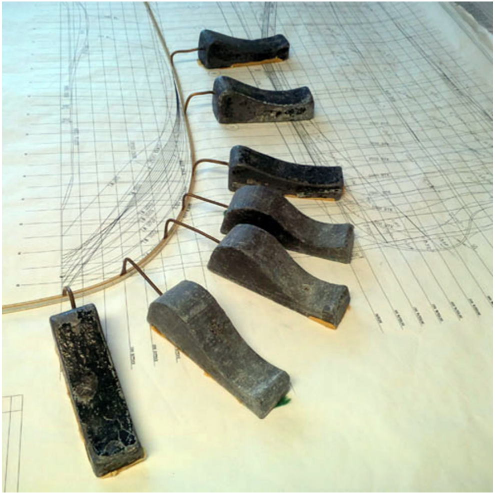
$$
\begin{array}{lll}
b_1(X) = 1, & b_3(X) = X^2, & b_5(X) = (X - \tau_1)_+^3 \\
b_2(X) = X, & b_4(X) = X^3, & b_6(X) = (X - \tau_2)_+^3
\end{array}
$$

Easy to show this is indeed a representation of a cubic spline (write out, show that in each knot, the relevant pair of functions has identical function values, and identical first and second derivatives.)

Cubic splines

“It is claimed that cubic splines are the lowest-order spline for which the knot-discontinuity is not visible to the human eye. There is seldom any good reason to go beyond cubic-splines, unless one is interested in smooth derivatives.” HTF page 144

Physical splines



Buy your own 'ducks' (knots): https://edsonmarine.com/products/home-accessories/spline-weights/

Back to the check list

What are we doing: Bayesian Penalized B-Splines Regression

1. What are splines?
2. What are B-splines?
3. How to fit B-Splines regression?
4. How are they penalized?
5. How to fit P-Splines in a Bayesian framework?

An alternative basis representation

- The previous representation of splines is called the **truncated power basis**
- There are many equivalent bases for representing splines
- The truncated power basis is conceptually simple, but not great numerically (we want to avoid potentially taking powers of large numbers)

Alternative: **B-spline basis**

B-spline basis

- Define a set of knot points $\tau_1 < \tau_2 < \ldots < \tau_{K+1}$
- Denote by $B_{i,m}(x)$ the ith B-spline basis function of degree m for the knot sequence τ
- The B-splines are defined recursively

For degree 0:

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } \tau_i \leq x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

Then

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m+1} - x}{\tau_{i+m+1} - \tau_{i+1}} B_{i+1,m-1}(x)$$

B-splines

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } \tau_i \leq x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

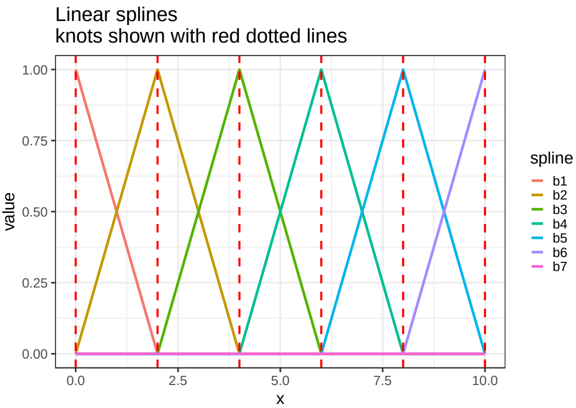
Then

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m+1} - x}{\tau_{i+m+1} - \tau_{i+1}} B_{i+1,m-1}(x)$$

- $B_{i,0}(x)$ is piecewise constant, indicating which knot span x is in
- In the recursion formula, the first part ramps up as x goes from τ_i to τ_{i+m}
- The second part ramps down as x goes from τ_{i+1} to τ_{i+m+1}

Linear B-splines

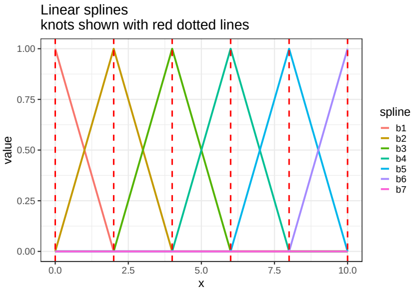
- For example, $B_{i,1}(x)$ is a triangular function that is zero below $x = \tau_i$, ramps to one at $x = \tau_{i+1}$ and back to zero at and beyond $x = \tau_{i+2}$.



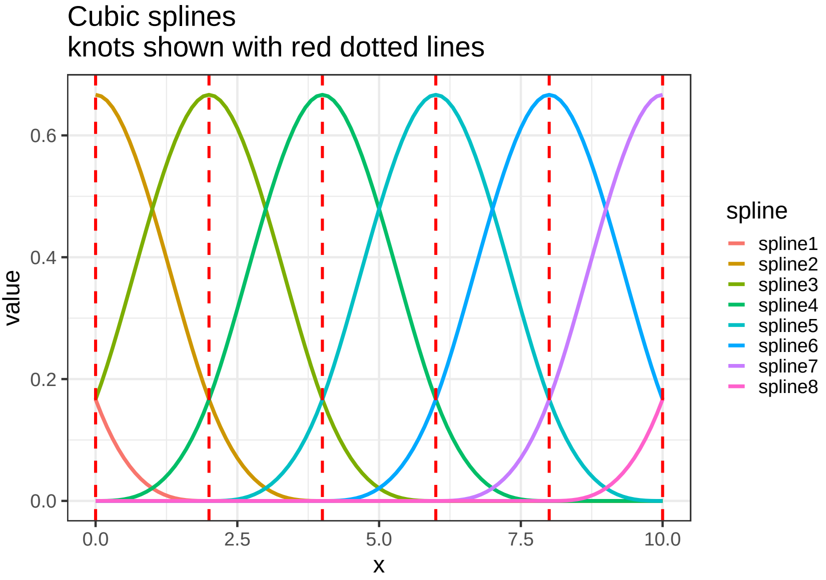
Linear B-splines

For linear splines

- Splines consists of 2 pieces, each of degree 1
- the pieces join at 1 knot
- Splines are non-zero on a domain spanning 3 knots
- Except at boundaries, overlap with 2 of their neighbors



Cubic B-splines



B-Splines

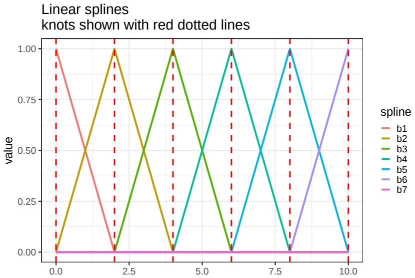
In general, a d-degree B-Spline has the general properties

- consists of $d + 1$ polynomial pieces, each of degree d
- the polynomial pieces join at d inner knots
- at the joining points, derivatives up to order $d - 1$ are continuous
- the B-spline is positive on a domain spanned by $d + 2$ knots; everywhere else it is zero;
- except at the boundaries, it overlaps with $2d$ polynomial pieces of its neighbors;
- at a given x, $d + 1$ B-splines are nonzero
- at any given x, $\sum_{k=1}^{K} B_k(x) = 1$

Generating B-Splines in R

Can just use bs function as part of the splines package as a starting point. e.g. the linear splines plot

```
library(splines)
x <- seq(0,10, by = 0.1)
knots <- seq(0,10, by = 2)
deg1 <- bs(x, degree = 1, knots = knots)
colnames(deg1) <- paste0("b", 1:(length(knots)+1))
deg1 <- as_tibble(deg1)
deg1 %>%
  mutate(x = x) %>%
  pivot_longer(-x, names_to = "spline", values_to = "value") %>%
  ggplot(aes(x, value, color = spline)) +
  geom_line(lwd = 1.5) +
  geom_vline(xintercept = knots, color = "red", lty = 2, lwd = 1.2)+
  theme_bw(base_size = 20) +
  ggtitle("Linear splines \nknots shown with red dotted lines")
```



Back to the check list

What are we doing: Bayesian Penalized B-Splines Regression

1. What are splines?
2. What are B-splines?
3. How to fit B-Splines regression?
4. How are they penalized?
5. How to fit P-Splines in a Bayesian framework?

How does this help us?

Model outcome y as

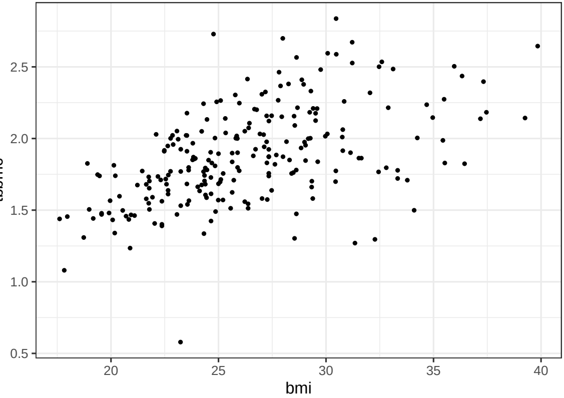
$$y_i = \mu\left(x_i\right) + \varepsilon_i$$
$$\varepsilon_i \sim N\left(0, \sigma_y^2\right)$$

with $\mu(x_i) = \sum_{k=1}^{K} B_k(x_i)\alpha_k$.

- choose knot points k, degree of splines
- generate basis splines B_k based on x_i's, fit regression to get estimates of α_k's
- Different α_k's give different $\mu(x)$s

Example (Lesaffre and Lawson (2012) Chapter 10)

- Modeling the relationship between osteoporosis and BMI
- Osteoporosis is a disease where the bone mineral density is reduced so that the bones become fragile.
- Marker used is total body bone mineral content (TBBMC, in kg).



Fit B-Splines regression

fit the model

$$y_i = \mu(x_i) + \varepsilon_i$$
$$\mu(x_i) = \sum_{k=1}^{K} B_k(x_i)\alpha_k$$
$$\varepsilon_i \sim N\left(0, \sigma_y^2\right)$$

where y is TBBMC and x is BMI. Let's use equally-spaced knots every 2.5 increments of BMI

Get B-Splines

```
I <- 2.5 # between-knot length
res <- getsplines(x.i, I = I) # a function to get splines of constant shape
B.ik <- res$B.ik
x.i[1] # look at first value of BMI
```

```
## [1] 23.61275
```

```
round(B.ik[1,],2) # value of splines at x.i[1]
```

```
##  spline1  spline2  spline3  spline4  spline5  spline6  spline7  spline8
##     0.00     0.00     0.00     0.16     0.67     0.17     0.00     0.00
##  spline9 spline10 spline11 spline12 spline13
##     0.00     0.00     0.00     0.00     0.00
```

```
sum(B.ik[1,]) # confirm = 1
```

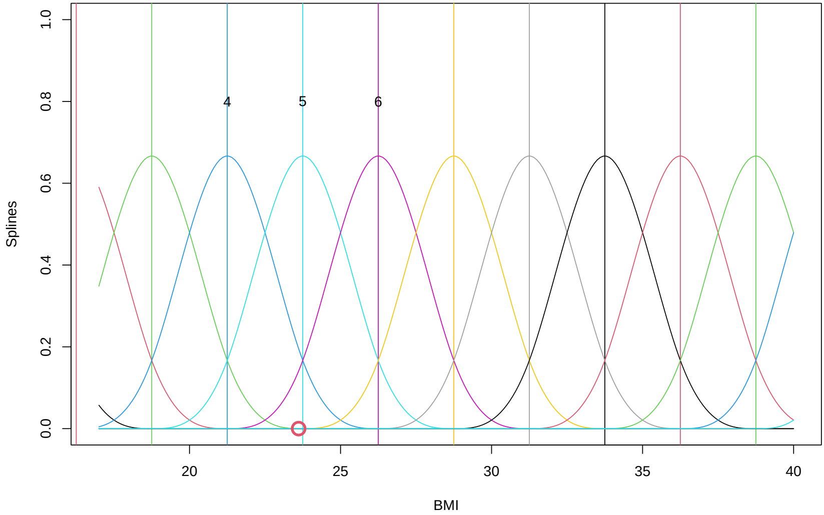
```
## [1] 1
```

```
round(res$knots.k,1) # indicates where spline is at max
```

```
##  spline1  spline2  spline3  spline4  spline5  spline6  spline7  spline8
##     13.6     16.1     18.6     21.1     23.6     26.1     28.6     31.1
##  spline9 spline10 spline11 spline12 spline13
##     33.6     36.1     38.6     41.1     43.6
```

Get B-Splines

red dot is x_1



Fit in Stan

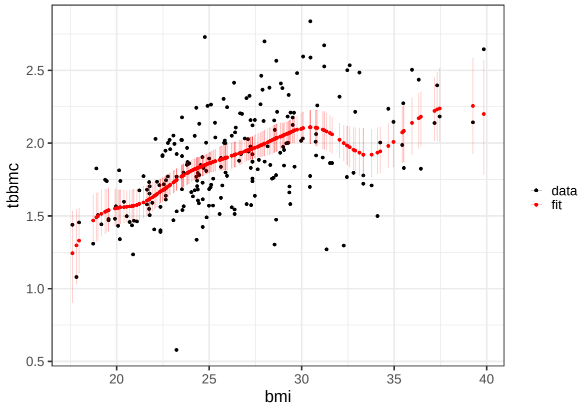
```
data {
  int<lower=0> N;
  int<lower=0> K;
  vector[N] y;
  matrix[N,K] B;
}
parameters {
  vector[K] alpha;
  real<lower=0> sigma;
}
transformed parameters{
  vector[N] mu;
  mu = B*alpha;
}
model {

   //likelihood
   y ~ normal(mu, sigma);
   //priors
   alpha ~ normal(0,1);
   sigma ~ normal(0,1);
}
```

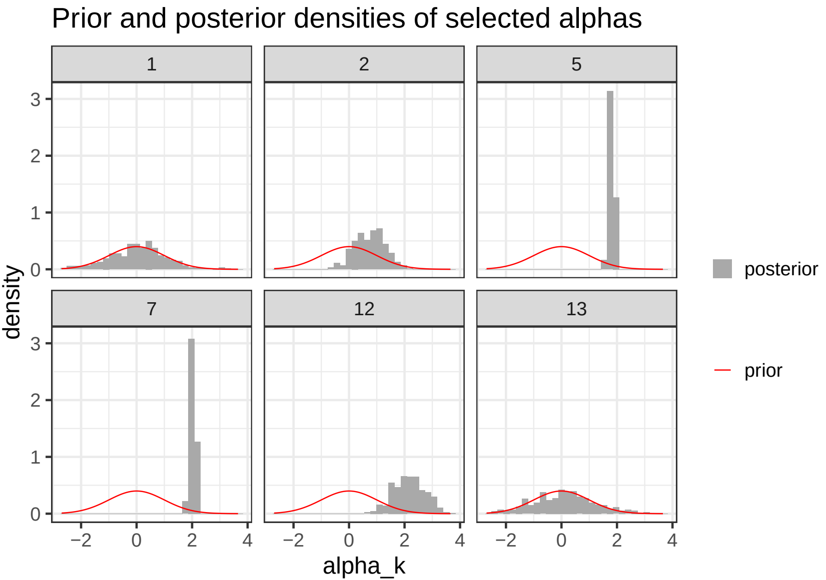
Results

Note, a good example of where gather_draws and median_qi is quick and easy

```
mod1 %>%
  gather_draws(mu[condition]) %>%
  median_qi() %>%
  ungroup() %>%
  mutate(tbbmc = y.i, bmi = x.i) %>%
  ggplot(aes(bmi, tbbmc)) + geom_point(aes(color = "data")) +
  geom_point(aes(bmi, .value,  color = "fit")) +
  geom_errorbar(aes(ymin = .lower, ymax = .upper, color = "fit"), alpha = 0.2) +
  scale_color_manual(name = "", values = c("fit" = "red", "data" = "black")) + theme_bw(base_size = 20)
```



The closer to the boundary, the more the α_k are informed by prior



Obtaining estimates for all x

- Can we estimate $\mu(x)$ for some x that was not in the data set, e.g. $x = 36$
- Yes, we can evaluate $\mu(x)$ at a grid of x values (as long as they are in the range of observed x values (such that α_k's have been estimated).
- Get posterior samples using posterior samples for α_k's

$$\mu(x)^{(s)} = \sum_{k=1}^{K} B_k(x)\alpha_k^{(s)}$$

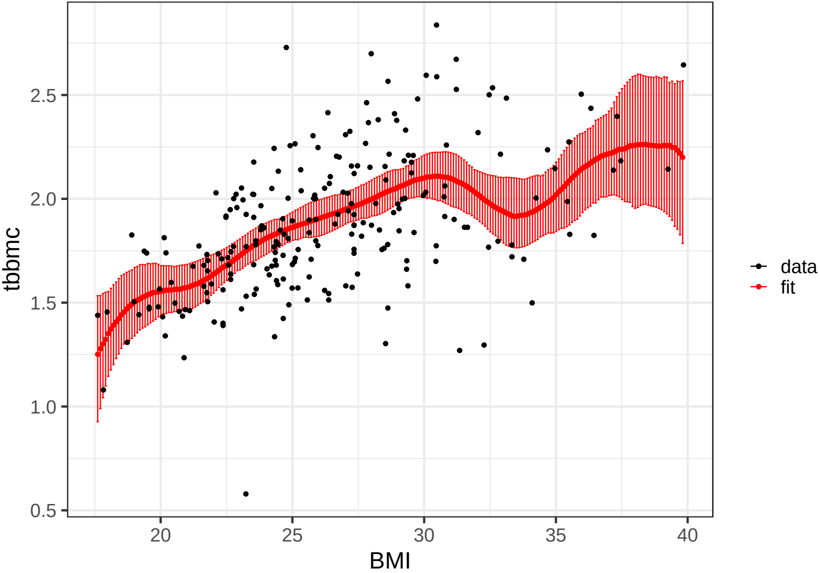
As usual, could do this in Stan or in R.

Obtaining estimates for "all" x

- Need to get values of B-Splines at each grid point then we are good to go

```
res_full <- getsplines(seq(min(x.i),max(x.i), by = 0.1), I = I)
B.ik_full <- res_full$B.ik #splines at full grid
alpha.sk <- extract(mod1)[["alpha"]] #post samples of alpha
CI.iq <- matrix(NA, nrow(B.ik_full),3)
for (i in 1:nrow(CI.iq)){
  CI.iq[i,] <- quantile(B.ik_full[i,]%*%t(alpha.sk),
                        c(0.025, 0.5, 0.975))
}
```

Obtaining estimates for "all" x



Note: what's the difference between generating new $\mu(x)$ versus new y

Back to the check list

What are we doing: Bayesian Penalized B-Splines Regression

1. What are splines?
2. What are B-splines?
3. How to fit B-Splines regression?

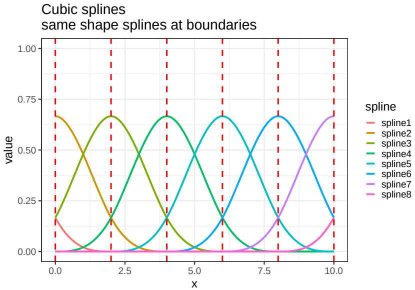
4. How are they penalized?

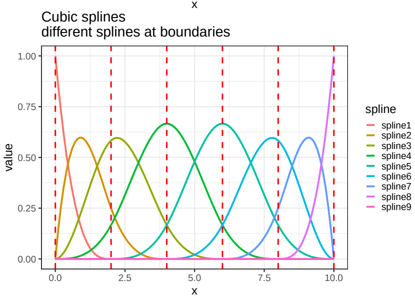
5. How to fit P-Splines in a Bayesian framework?

Choices

1. What degree splines should we use to fit (i.e. what should d be)?
   - in general, as mentioned above, should be at most cubic ($d = 3$)
2. What to do at boundaries
   - Above I used the same splines at the boundaries, resulting in some splines which overlap with the data range only partially.
   - This is easier to interpret when we get into penalization
   - Alternative implementations include fewer splines with higher values at the boundaries.

Choices


Cubic splines
same shape splines at boundaries
value
x
0.00
0.25
0.50
0.75
1.00
0.0
2.5
5.0
7.5
10.0
spline
spline1
spline2
spline3
spline4
spline5
spline6
spline7
spline8


Cubic splines
different splines at boundaries
value
x
0.00
0.25
0.50
0.75
1.00
0.0
2.5
5.0
7.5
10.0
spline
spline1
spline2
spline3
spline4
spline5
spline6
spline7
spline8
spline9

Choices

3. Where to put the knots?

- Options include equally spaced or based on data availability (e.g., based on percentiles of data set)
- I prefer equally spaced splines
  - to avoid limited uncertainty for x-values where data are sparse
  - to facilitate interpretation when a penalization is used
  - ... but as usual, sensitivity to model choice (in this case, knot placement) should be checked

4. How many knots?

- The regression is most flexible in regions that contain a lot of knots, because in those regions the coefficients can change rapidly
- Distance needs to be small enough to capture (true) fluctuations in outcome of interest ($\mu(x)$)
- But too many knots may be just picking up random fluctuations

Smoothing penalties

- In fitting a smooth curve to a set of data, we want the residual sum of squares to be small

$$\text{RSS}(f) = \sum_{i=1}^{n} \{y_i - f(x_i)\}^2$$

where $f(x_i) = \sum_{k=1}^{K} \alpha_k B_k(x_i)$.

- But if we don't put any constraints on $\sum_{k=1}^{K} \alpha_k B_k(x_i)$, we can make RSS zero by just interpolating all the points
- If the number of knots be relatively large, such that the fitted curve will show more variation than is justified by the data
- What we really want is a function that makes S small but is also **smooth**
- Add another term that penalizes fluctuations in spline

Smoothing splines

- Penalized RSS (O'Sullivan, Hastie)

$$\mathrm{RSS}(f, \lambda) = \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 \, dt$$

where λ is a fixed smoothing parameter. Loss+penalty / bias v variance trade-off:

- The smaller the λ, the closer f is to interpolation
- As $\lambda \to \infty$, we get a simple least squares line fit.
- $\int \{f''(t)\}^2 \, dt$ a measure of total change in the derivative
- The bigger $\int \{f''(t)\}^2 \, dt$, the more 'rough' the fit
- It can be shown that the function that minimizes this penalized RSS is a natural cubic spline (cubic spline + linear in the region outside the extreme knots) with knots at each x_i. Then choose λ based on LOO-CV. (HTF Chap 5)

Alternative: P-Splines

Eilers and Marx (1996) suggested an alternative smoothing penalty:

$$\text{RSS}(f, \lambda) = \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 + \lambda \sum_{k=q+1}^{K} (\Delta^q \alpha_k)^2$$

- the penalty is based on finite differences of the coefficients of adjacent B-splines
- This reduces the dimensionality of the problem to K, the number of B-splines, instead of n, the number of observations
- A good discrete approximation to term used in the previous slide
- q could be anything, in practice usually consider penalizing first- or second-order differences
- These are called penalized B-splines, or P-splines

P-splines

In likelihood-based inference, want to maximize the penalized likelihood

$$L = \ell\left(y, \alpha_1, \ldots, \alpha_k, \right) - \lambda \sum_{k=q+1}^{K} \left(\Delta^q \alpha_k\right)^2$$

This leads to the system of equations

$$B^{'}(y - \mu) = \lambda D_q^{'} D_q \alpha$$

- D_q is the matrix representation of the difference operator, e.g. D_1 is dimensions $(k-1) \times k$ and has elements $d_{ij} = -1$ if $i = j$, $d_{ij} = 1$ if $i = j - 1$ and 0 otherwise
- Elements of B are $B_k(x_i)$
- Note if $\lambda = 0$, just usual linear regression with B-spline basis. If $q = 0$, this is ridge regression
- For fixed D_q and λ, can solve using IWLS
- Choose smoothing parameter λ based on AIC or CV

Back to the check list

What are we doing: Bayesian Penalized B-Splines Regression

1. What are splines?
2. What are B-splines?
3. How to fit B-Splines regression?
4. How are they penalized?

5. How to fit P-Splines in a Bayesian framework?

Bayesian P-Splines

Our penalized likelihood is

$$L = l\left(y, \alpha_1, \ldots, \alpha_k, \right) - \lambda \sum_{k=q+1}^{K} \left(\Delta^q \alpha_k\right)^2$$

- In the Bayesian context, the spline coefficients α_k are now random variables, which need prior distributions
- The stochastic equivalent to the difference penalties is to place a q-order random walk prior on the α_ks

Bayesian P-Splines

- E.g. for a first-order difference penalty, the prior is

$$\alpha_k = \alpha_{k-1} + \varepsilon_k$$
$$\varepsilon_k | \sigma \sim N\left(0, \sigma_\alpha^2\right)$$

  or equivalently

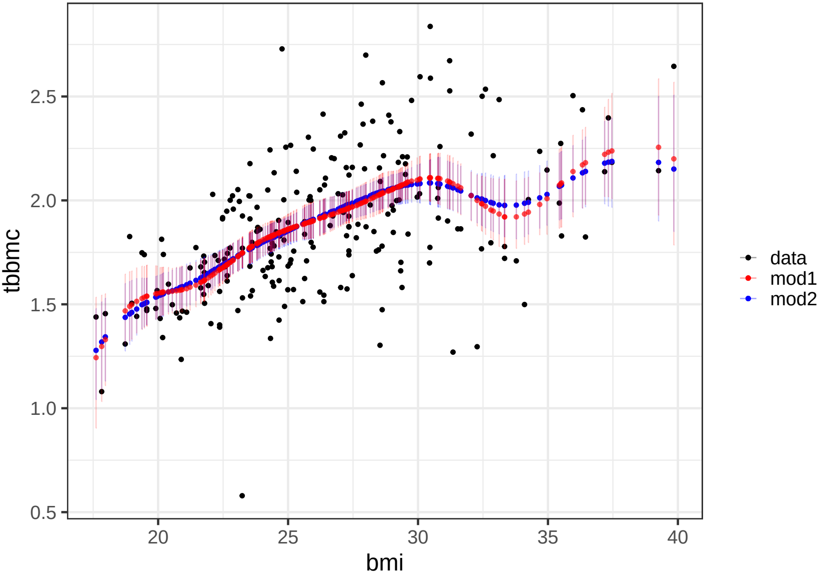
$$\alpha_k \sim N(\alpha_{k-1}, \sigma_\alpha^2)$$

  (We saw these last week in the context of temporal models!)
- The σ_α^2 term is equivalent to λ, and controls the smoothness of fit
- The smaller σ_α^2, the smoother the fit

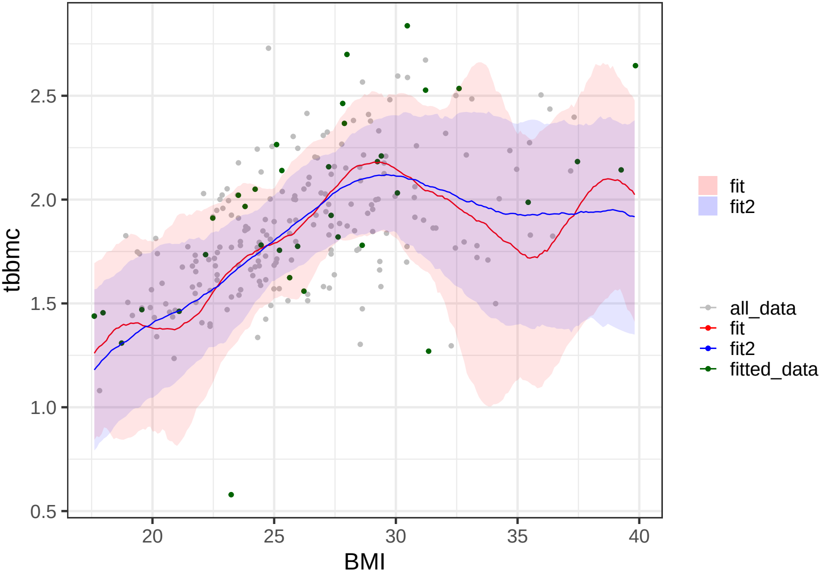
How to fit in Stan

```
data {
  int<lower=0> N;
  int<lower=0> K;
  vector[N] y;
  matrix[N,K] B;
}
parameters {
  vector[K] alpha;
  real<lower=0> sigma;
  real<lower=0> sigma_alpha;
}
transformed parameters{
  vector[N] mu;
  mu = B*alpha;
}
model {
   //likelihood
   y ~ normal(mu, sigma);
   //priors
   alpha[1] ~ normal(0, sigma_alpha);
   alpha[2:K] ~ normal(alpha[1:(K - 1)], sigma_alpha);
   alpha ~ normal(0,1);
   sigma ~ normal(0,1);
   sigma_alpha ~ normal(0,1);
}
```

Example



But what if we had a lot less data

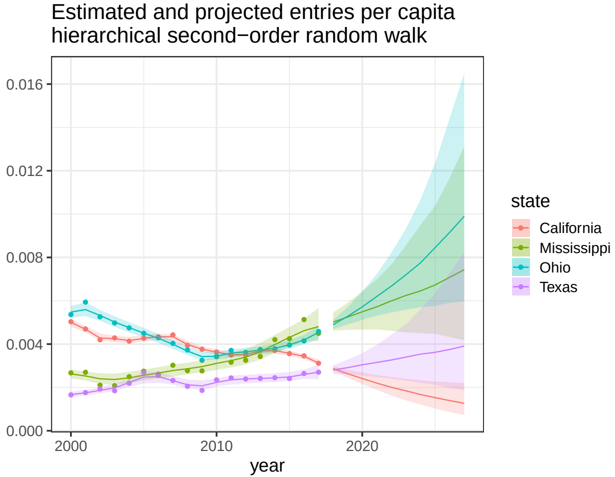


... and second-order penalization would be even more smooth.

P-splines for temporal smoothing

Temporal smoothing

- Last week we talked about models to fit and project time series
- E.g. we fit a (hierarchical) RW2 model to estimate and project foster care populations in the US



Temporal smoothing

- We could have used P-splines in same way
- Fit a model with second-order P-splines

$$y_{st} \sim N(\log \lambda_{st}, \sigma_y^2)$$
$$\log \lambda_{st} = \alpha_k B_k(t)$$

with

$$\Delta^2 \alpha_k \sim N(0, \sigma_{\alpha,s}^2)$$

- $y_{st} = \log(E_{st}/P_{st})$, E is entries, P is population
- For comparison, the model we fit last week was essentially $\Delta^2 \log \lambda_{st} \sim N(0, \sigma_\lambda^2)$.
- RW2 process now on coefficients, not data
- Projection happens through projection of the coefficients

Temporal smoothing

$$y_{st} \sim N(\log \lambda_{st}, \sigma_y^2)$$
$$\log \lambda_{st} = \alpha_k B_k(t)$$

with

$$\Delta^2 \alpha_k \sim N(0, \sigma_{\alpha,s}^2)$$

In addition, want to model variance hierarchically

$$\log \sigma_{\alpha,s} \sim N(\mu_\sigma, \tau^2)$$

- with P-splines, usually choose to have a relatively large number of splines (knot points, k), then smooth away fluctuations
- I ran model with knots every 2.5 years, with constant spacing across all states and boundary splines that are same shape
- Helps to interpret σ_α's as smoothing parameters, given the same spline settings

How to get P-spline projections?

In R to spell it out (could do in Stan)

```
proj_years <- 2018:2030

# Note: B.ik are splines for in-sample period
# has dimensions i (number of years) x k (number of knots)

# need splines for whole period
B.ik_full <- getsplines(c(years, proj_years))$B.ik

K <- ncol(B.ik) # number of knots in sample
K_full <- ncol(B.ik_full) # number of knots over entire period
proj_steps <- K_full - K # number of projection steps

# get your posterior samples
alphas <- extract(mod)[["alpha"]]
sigmas <- extract(mod)[["sigma"]] # sigma_alpha
sigma_ys <- extract(mod)[["sigma_y"]]

nsims <- nrow(alphas)
```

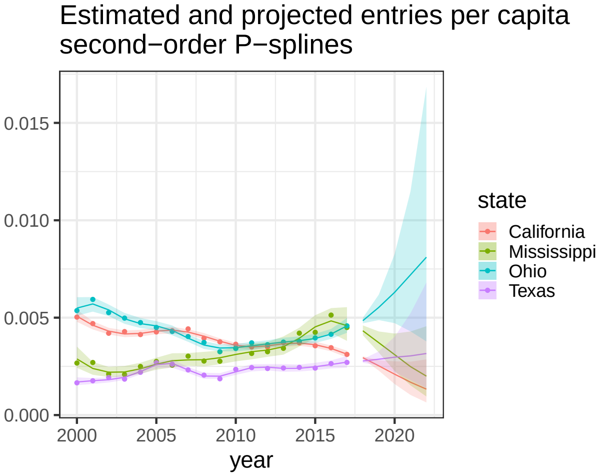
```r
# first, project the alphas
alphas_proj <- array(NA, c(nsims, proj_steps, length(states)))
set.seed(1098)
# project the alphas
for(j in 1:length(states)){
    first_next_alpha <- rnorm(n = nsims,
                              mean = 2*alphas[,K,j] - alphas[,K-1,j],
                              sd = sigmas[,j])
    second_next_alpha <- rnorm(n = nsims,
                               mean = 2*first_next_alpha - alphas[,K,j],
                               sd = sigmas[,j])

    alphas_proj[,1,j] <- first_next_alpha
    alphas_proj[,2,j] <- second_next_alpha
    # now project the rest
    for(i in 3:proj_steps){ #!!! not over years but over knots
      alphas_proj[,i,j] <- rnorm(n = nsims,
                                 mean = 2*alphas_proj[,i-1,j] - alphas_proj[,i-2,j],
                                 sd = sigmas[,j])
    }
}

# now use these to get y's
y_proj <- array(NA, c(nsims, length(proj_years), length(states)))
for(i in 1:length(proj_years)){ # now over years
  for(j in 1:length(states)){
    all_alphas <- cbind(alphas[,,j], alphas_proj[,,j] )
    this_lambda <- all_alphas %*% as.matrix(B.ik_full[length(years)+i, ])
    y_proj[,i,j] <- rnorm(n = nsims, mean = this_lambda, sd = sigma_ys[,j])
  }
}

# then proceed as normal to get median, quantiles etc
```

P-splines projection



Temporal take-aways

- From a time series perspective, P-splines are a nice way of capturing non-linear trends over time
- Temporal structure is on coefficients (from knot point to knot point) rather than on data (from time point to time point)
- Can hierarchically smooth fluctuations in the same way we did with temporal models last week
- Particularly powerful in combination with covariates (outcome = expected + smoothed fluctuations)
- Compared to random walks or ARIMA processes, will have different uncertainty in fit and projections
- Not much different in foster care case, but becomes more obvious when data are sparse