Week 11: Splines

04/04/23

Overview

In this lab you'll be fitting a second-order P-Splines regression model to foster care entries by state in the US, projecting out to 2030.

```r
library(tidyverse)
library(here)
library(rstan)
library(tidybayes)
source("code/getsplines.R")
```

Here's the data

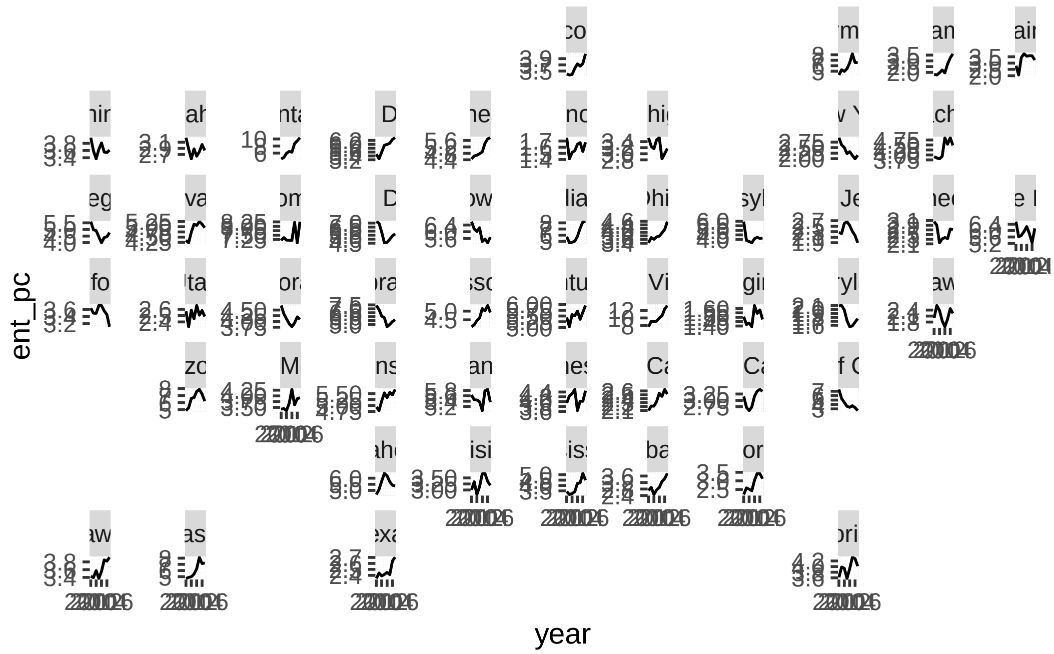
```r
d <- read_csv("data/fc_entries.csv")
```

Question 1

Make a plot highlighting trends over time by state. Might be a good opportunity to use `geofacet`. Describe what you see in a couple of sentences.

```r
library(geofacet)

d |>
  ggplot(aes(year, ent_pc)) +
  geom_line() +
  facet_geo(~state, scales = 'free_y')
```



Question 2

Fit a hierarchical second-order P-Splines regression model to estimate the (logged) entries per capita over the period 2010-2017. The model you want to fit is

$$\begin{aligned} y_{st} &\sim N(\log \lambda_{st}, \sigma^2_{y,s}) \\ \log \lambda_{st} &= \alpha_k B_k(t) \\ \Delta^2 \alpha_k &\sim N(0, \sigma^2_{\alpha,s}) \\ \log \sigma_{\alpha,s} &\sim N(\mu_\sigma, \tau^2) \end{aligned}$$

Where $y_{s,t}$ is the logged entries per capita for state s in year t. Use cubic splines that have knots 2.5 years apart and are a constant shape at the boundaries. Put standard normal priors on standard deviations and hyperparameters.

```
years <- unique(d$year)
N <- length(years)
y <- log(d |>
  select(state,year, ent_pc) |>
  pivot_wider(names_from = 'state', values_from = 'ent_pc') |>
  select(-year) |>
  as.matrix())
```

```r
res <- getsplines(years, 2.5)
B <- res$B.ik
K <- ncol(B)

stan_data <- list(N = N, y =y , K= K, S = length(unique(d$state)), B =B)
mod <- stan(data = stan_data, file= 'l11q2.stan')
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Libra
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/library/RcppEigen/include/Eigen/
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'l11q2' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000135 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.35 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 11.2333 seconds (Warm-up)
Chain 1:                7.22958 seconds (Sampling)
Chain 1:                18.4628 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'l11q2' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000109 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.09 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 11.3262 seconds (Warm-up)
Chain 2:                8.53654 seconds (Sampling)
Chain 2:                19.8628 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'l11q2' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000116 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.16 seconds.
Chain 3: Adjust your expectations accordingly!
```

```
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 11.2025 seconds (Warm-up)
Chain 3:                7.82532 seconds (Sampling)
Chain 3:                19.0279 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'l11q2' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.00011 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.1 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 11.6661 seconds (Warm-up)
Chain 4:                7.98039 seconds (Sampling)
Chain 4:                19.6464 seconds (Total)
```

Chain 4:

Question 3

Project forward entries per capita to 2030. Pick 4 states and plot the results (with 95% CIs). Note the code to do this in R is in the lecture slides.

```
states <- unique(d$state)
B.ik <- B
I <- 2.5
proj_years <- 2018:2030
# Note: B.ik are splines for in-sample period
# has dimensions i (number of years) x k (number of knots)
# need splines for whole period
B.ik_full <- getsplines(c(years, proj_years), I)$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_alpha"]] # sigma_alpha
sigma_ys <- extract(mod)[["sigma_y"]]
nsims <- nrow(alphas)

# 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
```

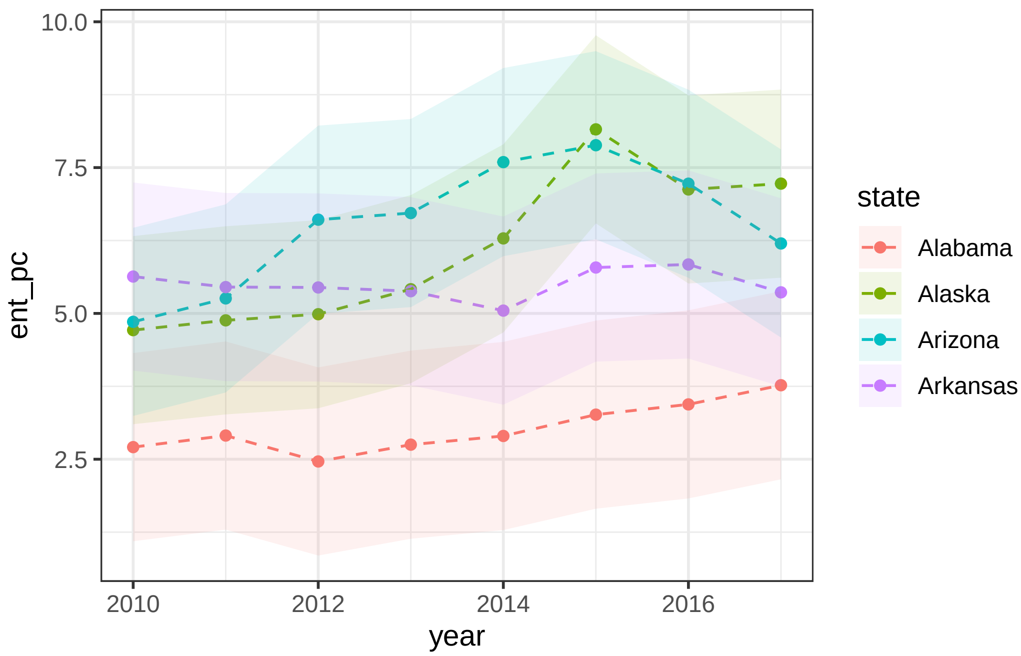
```
alabama <- y_proj[, , 1] |>
  as_tibble() |>
  set_names(2018:2030) |>
  median_qi() |>
  pivot_longer(`2018`:`2030`, names_to = 'year')

alaska <- y_proj[, , 2] |>
  as_tibble() |>
  set_names(2018:2030) |>
  median_qi()

arizona <- y_proj[, , 3] |>
  as_tibble() |>
  set_names(2018:2030) |>
  median_qi()


arkansas <- y_proj[, , 4] |>
  as_tibble() |>
  set_names(2018:2030) |>
  median_qi()
```

```r
d |>
  filter(state %in% c("Alabama", "Alaska","Arizona", "Arkansas")) |>
  ggplot(aes(year, ent_pc)) +
  geom_point(aes( color = state)) +
  geom_line(aes( color = state), lty = 2) +
  geom_ribbon(aes(ymin = ent_pc - sd(ent_pc),
                  ymax = ent_pc + sd(ent_pc),
                  fill =  state), alpha = 0.1) +
  theme_bw()
```



Question 4 (bonus)

P-Splines are quite useful in structural time series models, when you are using a model of the form

$$f(y_t) = \text{systematic part} + \text{time-specific deviations}$$

where the systematic part is model with a set of covariates for example, and P-splines are used to smooth data-driven deviations over time. Consider adding covariates to the model you ran above. What are some potential issues that may happen in estimation? Can you think of an additional constraint to add to the model that would overcome these issues?