Week 11: Splines

30/03/23

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.

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

Here's the data

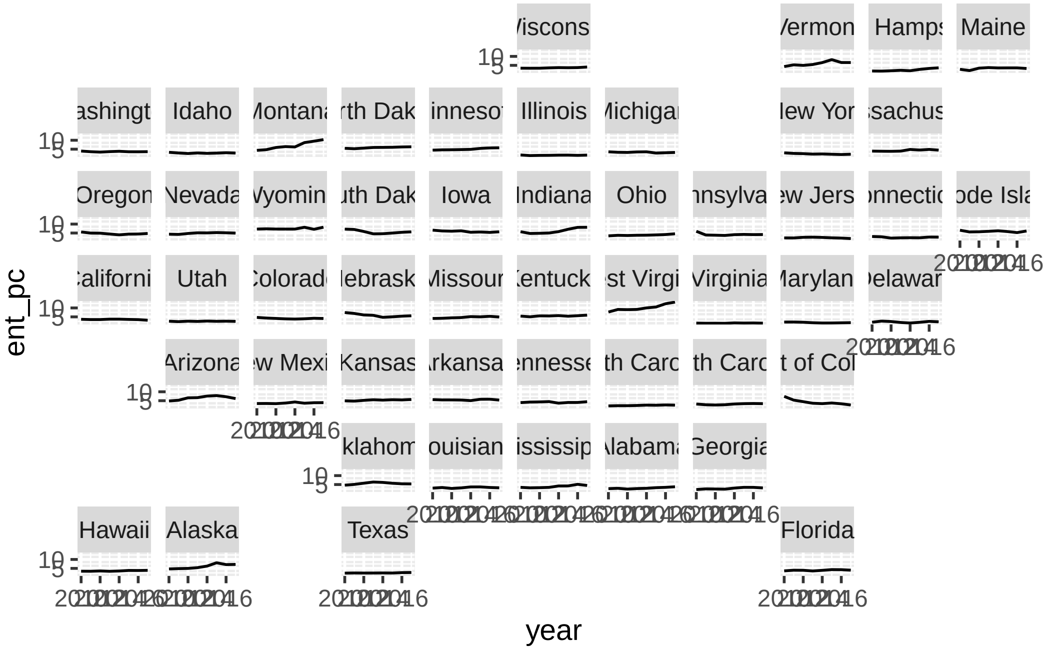
```
d <- read_csv(here("data/fc_entries.csv"))
```

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.

```
library(geofacet)
```

```
d %>%
  ggplot(aes(year, ent_pc)) +
  geom_line() +
  facet_geo(~state)
```



State-wide variation is relatively low since the same scale plot shows similar plots for each state. There are some exceptions however. I see that Montana has high entries relative to other states, and so does Alaska and Arizona. This is interesting, as I am not sure why foster care entries per capita would be higher in these states. I also see states like New York and California, which tend to be more subsidized socially have lower foster care per capita rates. The district of Columbia shows a decreasing trend.

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.

```r
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())

res <- getsplines(years, 2.5, degree = 3)
B <- res$B.ik
K <- ncol(B)
S <- length(unique(d$state))
```

```r
stan_data <- list(N = N, y=y, K = K, S = length(unique(d$state)),
                  B = B)
```

```r
mod <- stan(data = stan_data,
            file = here("code/models/lab11_1.stan"),
            iter = 2000)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000428 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.28 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: 19.632 seconds (Warm-up)
Chain 1:                12.399 seconds (Sampling)
Chain 1:                32.031 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000152 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.52 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: 20.845 seconds (Warm-up)
Chain 2:                10.582 seconds (Sampling)
Chain 2:                31.427 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000117 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.17 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: 19.075 seconds (Warm-up)
Chain 3:                12.985 seconds (Sampling)
Chain 3:                32.06 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000113 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.13 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: 18.363 seconds (Warm-up)
Chain 4:                14.742 seconds (Sampling)
Chain 4:                33.105 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.

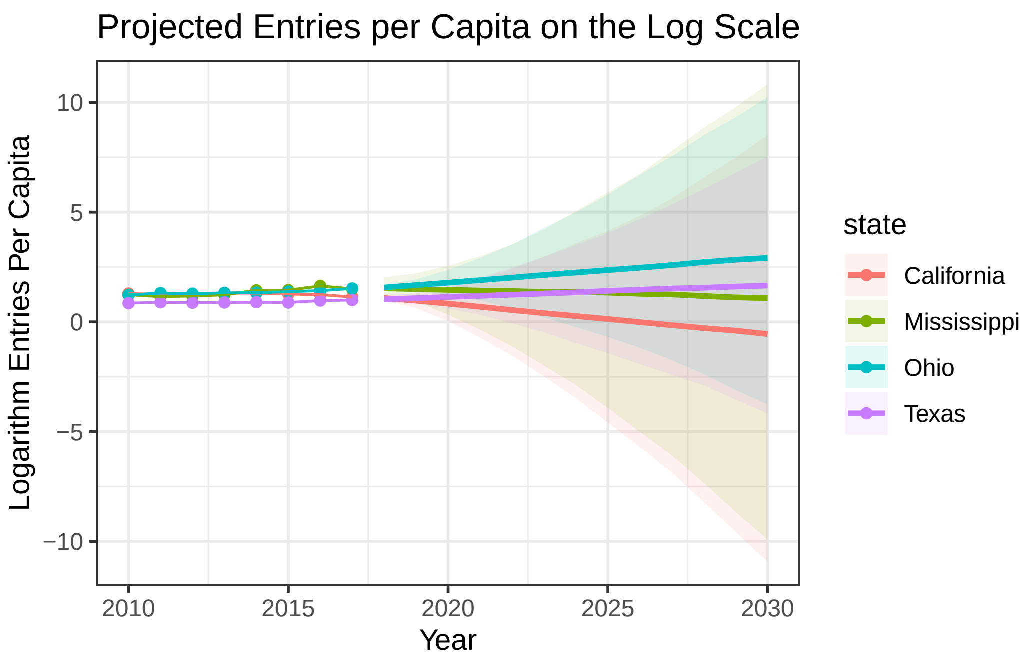
```
B.ik <- B
proj_years <- 2018:2030
# Note: B.ik are splines for in-sample period
# has dimensions i (number of years) x k (number of knots)
# need splines for whole period
B.ik_full <- getsplines(c(years, proj_years), 2.5, degree=3)$B.ik
K <- ncol(B.ik) # number of knots in sample
K_full <- ncol(B.ik_full) # 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, S))
set.seed(1098)
# project the alphas
for(j in 1:S){
  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), S))
for(i in 1:length(proj_years)){ # now over years
  for(j in 1:S){
    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])
  }
}
```

```
states = c("California", "Mississippi" , "Ohio", "Texas")
# Index for states
ind = which(unique(d$state) %in% states)

# Y proj for states
state_yproj = y_proj[,, ind]

state1 <- as.tibble(state_yproj[,,1])
state2 <- as.tibble(state_yproj[,,2])
state3 <- as.tibble(state_yproj[,,3])
state4 <-as.tibble(state_yproj[,,4])

p <- state1 %>%
  summarize(across(everything(), list(median = median,
                                      q2.5 = ~quantile(., probs = 0.025),
                                      q97.5 = ~quantile(., probs = 0.975)))) %>%
  pivot_longer(cols = everything(), names_to = c(".value", "column"), names_sep = "_")
p <- as.tibble(t(as.matrix(p)))
p <- p[2:nrow(p),]
colnames(p) <- c("med", "lower", "upper")

p2 <- state2 %>%
  summarize(across(everything(), list(median = median,
                                      q2.5 = ~quantile(., probs = 0.025),
                                      q97.5 = ~quantile(., probs = 0.975)))) %>%
  pivot_longer(cols = everything(), names_to = c(".value", "column"), names_sep = "_")
p2 <- as.tibble(t(as.matrix(p2)))
p2 <- p2[2:nrow(p2),]
colnames(p2) <- c("med", "lower", "upper")

p3 <- state3 %>%
  summarize(across(everything(), list(median = median, q2.5 = ~quantile(., probs = 0.025),
  pivot_longer(cols = everything(), names_to = c(".value", "column"), names_sep = "_")
p3 <- as.tibble(t(as.matrix(p3)))
p3 <- p3[2:nrow(p3),]
colnames(p3) <- c("med", "lower", "upper")
```

```
p4 <- state4 %>%
  summarize(across(everything(), list(median = median, q2.5 = ~quantile(., probs = 0.025),
  pivot_longer(cols = everything(), names_to = c(".value", "column"), names_sep = "_")
p4 <- as.tibble(t(as.matrix(p4)))
p4 <- p4[2:nrow(p4),]
colnames(p4) <- c("med", "lower", "upper")

p$year <- proj_years
p$state <- rep(states[1], nrow(p))

p2$year <- proj_years
p2$state <- rep(states[2], nrow(p))

p3$year <- proj_years
p3$state <- rep(states[3], nrow(p))

p4$year <- proj_years
p4$state <- rep(states[4], nrow(p))
```

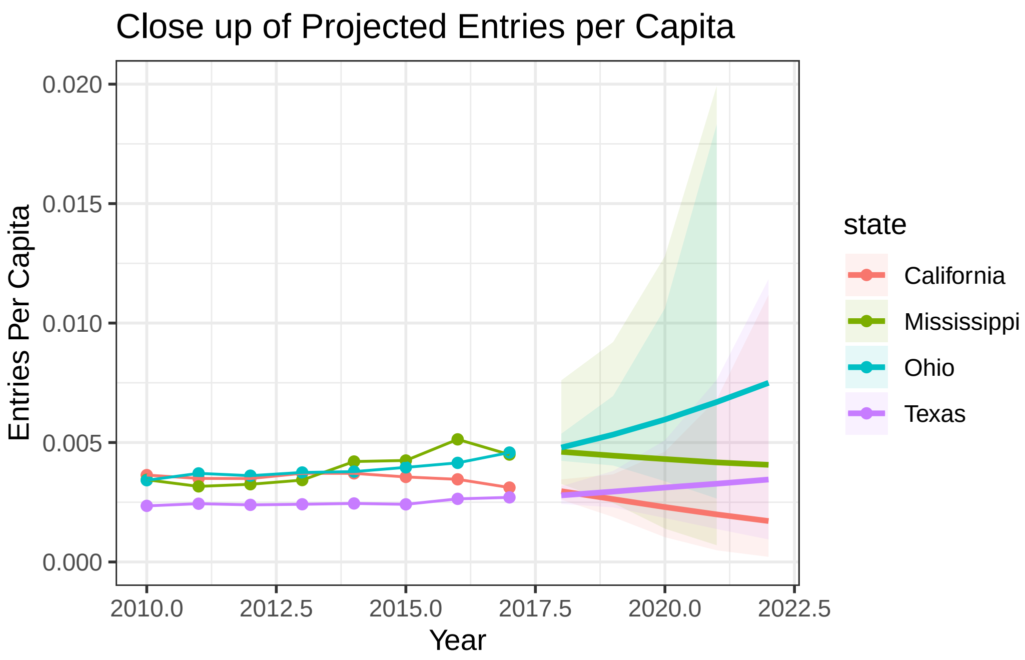
```
dat <- bind_rows(p, p2, p3, p4)
dat$med <- (as.numeric(dat$med))
dat$lower <- (as.numeric(dat$lower))
dat$upper <- (as.numeric(dat$upper))

d %>%
  filter(state %in% states) %>%
  ggplot(aes(x=year, color=state)) +
  geom_point(aes(y=log(ent_pc))) +
  geom_line(aes(y=log(ent_pc))) + theme_bw() +
  geom_ribbon(data=dat, aes(x=year, ymax=upper,
                          ymin=lower, fill = state), alpha = 0.1, colour = NA) +
  geom_line(data=dat, aes(x=year, y=med, color=state), linewidth = 1)  +
  xlab("Year") + ylab("Logarithm Entries Per Capita") +
  ggtitle("Projected Entries per Capita on the Log Scale")
```



```
 # xlim(2010,2025) +
#  ylim(0,0.01)
```

Here is a close-up on the normal scale (note it cuts off some of the shaded region):

```
dat <- bind_rows(p, p2, p3, p4)
dat$med <- exp(as.numeric(dat$med))/1000
dat$lower <- exp(as.numeric(dat$lower))/1000
dat$upper <- exp(as.numeric(dat$upper))/1000

d %>%
  filter(state %in% states) %>%
  ggplot(aes(x=year, color=state)) +
  geom_point(aes(y=(ent_pc)/1000)) +
  geom_line(aes(y=(ent_pc)/1000)) + theme_bw() +
  geom_ribbon(data=dat, aes(x=year, ymax=upper,
                            ymin=lower, fill = state), alpha = 0.1, colour = NA) +
  geom_line(data=dat, aes(x=year, y=med, color=state), linewidth = 1)  +
  xlab("Year") + ylab("Entries Per Capita") +
  ggtitle("Close up of Projected Entries per Capita") +
  xlim(2010,2022) +  ylim(0,0.02)
```



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?

Some potential issues are that the model will not converge. This is because with stationary models, everything converges to the mean and the variance is constant with time. With a random walk model where the time series is not stationary, then it is hard to determine whether the trend is due to the covariates or due to splines. In other words, it is hard to distinguish trends due to the covariates from trends due to the splines. To compensate for this, we can constrain our splines such that they sum to zero.