Lab 11

02/04/23

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(here("code/getsplines.R"))
```

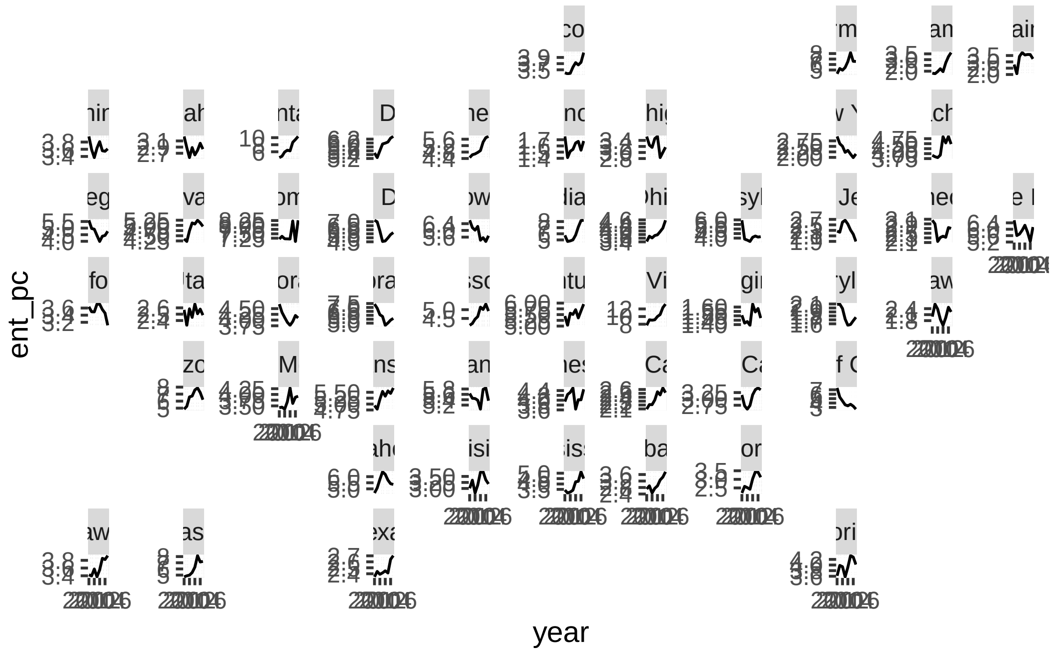
Here's the data

```r
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.

```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.ik <- res$B.ik
K <- ncol(B.ik)

stan_data <- list(N=N,y=y,K=K,S=length(unique(d$state)),B=B.ik)
mod <- stan(data = stan_data, file = "../code/models/lab11.stan")
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000167 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.67 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: 9.712 seconds (Warm-up)
Chain 1:                7.049 seconds (Sampling)
Chain 1:                16.761 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000101 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.01 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: 10.866 seconds (Warm-up)
Chain 2:                7.017 seconds (Sampling)
Chain 2:                17.883 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000101 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.01 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: 10.035 seconds (Warm-up)
Chain 3:                6.476 seconds (Sampling)
Chain 3:                16.511 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
```

```
Chain 4: Gradient evaluation took 0.000101 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.01 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: 10.307 seconds (Warm-up)
Chain 4:                6.306 seconds (Sampling)
Chain 4:                16.613 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.

```r
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)$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 <- rstan::extract(mod)[["alpha"]]
sigmas <- rstan::extract(mod)[["sigma_alpha"]] # sigma_alpha
sigma_ys <- rstan::extract(mod)[["sigma_y"]]
nsims <- nrow(alphas)
```

```r
states = unique(d$state)
# 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 = sigma
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
```

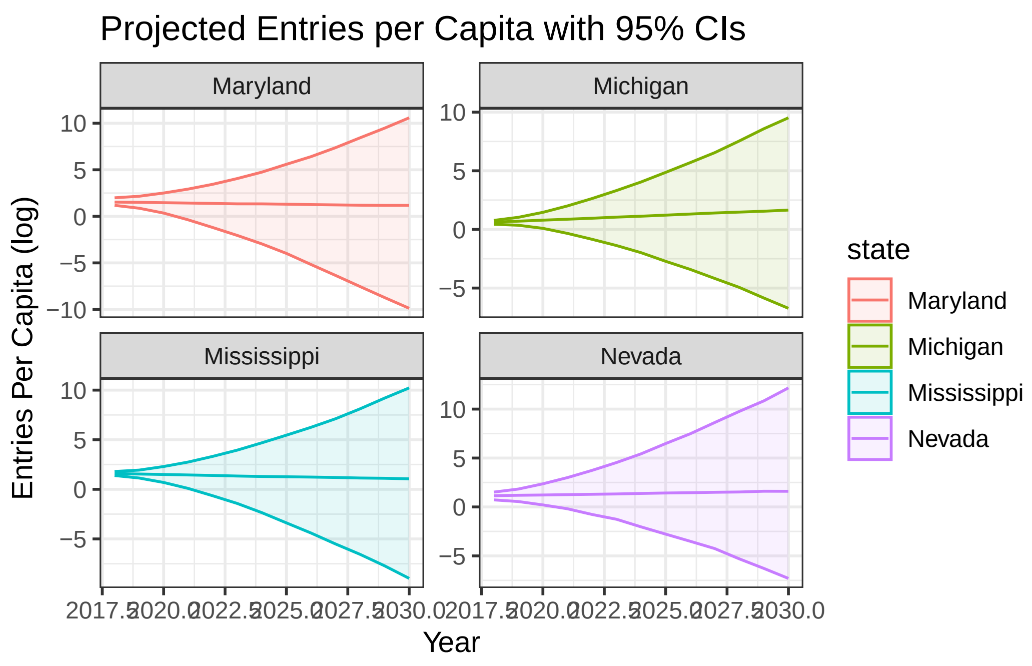
```r
# Choose four states
selected_states <- c("Michigan", "Nevada", "Maryland", "Mississippi")
state_indices <- which(unique(d$state) %in% selected_states)

# Extract projections for the chosen states
selected_yproj <- y_proj[, , state_indices]

# Calculate median and quantiles for each state
proj_data <- lapply(1:length(selected_states), function(j) {
  med <- apply(selected_yproj[, , j], 2, median)
  lower <- apply(selected_yproj[, , j], 2, function(x) quantile(x, 0.025))
  upper <- apply(selected_yproj[, , j], 2, function(x) quantile(x, 0.975))
  return(data.frame(year = proj_years,
                    state = selected_states[j],
                    median = med,
                    lower = lower,
                    upper = upper))
})

# Combine the data from all states
proj_data_combined <- do.call(rbind, proj_data)

# Plot the results with 95% CIs
ggplot(proj_data_combined, aes(x = year, y = median, color = state)) +
  geom_line() +
  geom_ribbon(aes(ymin = lower, ymax = upper, fill = state), alpha = 0.1) +
  theme_bw() +
  xlab("Year") +
  ylab("Entries Per Capita (log)") +
  ggtitle("Projected Entries per Capita with 95% CIs") +
  facet_wrap(~state, scales = "free_y", ncol = 2)
```



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?