Week 11: Splines

Xuanze Li

31/03/24

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("getsplines.R")
```

Here's the data

```
d <- read_csv("fc_entries.csv")
d <- d %>% na.omit()
```

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.

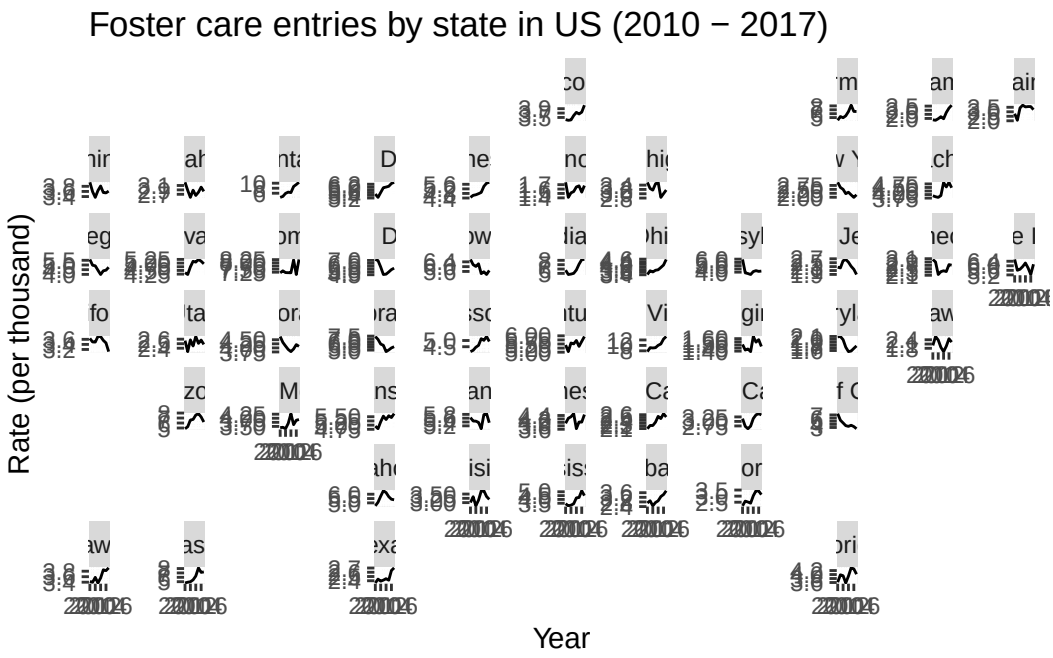
```
options(repos = c(CRAN = "https://cran.rstudio.com"))
install.packages("geofacet")
```

```
The downloaded binary packages are in
    /var/folders/jb/5jg6vnf908z8kxmvdfpvhqcw0000gn/T//RtmpSjPleK/downloaded_packages
```

```
install.packages("dplyr")
```

```
The downloaded binary packages are in
    /var/folders/jb/5jg6vnf908z8kxmvdfpvhqcw0000gn/T//RtmpSjPleK/downloaded_packages
```

```
library(geofacet)
library(ggplot2)
library(dplyr)

d %>%
  ggplot(aes(year, ent_pc))+
  geom_line()+
  facet_geo(~state, scale="free_y")+
  labs(title = "Foster care entries by state in US (2010 - 2017)",
    x = "Year",
    y = "Rate (per thousand)")
```



In this graph, we can see very different trends by state. Some states have trended upward from 2010 to 2017 (e.g., Ohio). Some states had ups and downs and generally trended down (e.g.,

Iowa). Some states declined and then rebounded (e.g., Colorado). Overall, more states saw an increase in home prices from 2010 to 2017. Each state has its own characteristics, so it would be more approporiate to consider modeling them separately (at the state level).

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.

```
library(dplyr)
library(tidyr)

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)
B <- res$B.ik
K <- ncol(B)
stan_data <- list(N=N, y = y, K=K, S = length(unique(d$state)), B=B)
```

```
library(rstan)
model <- stan(data = stan_data, file = "lab11.stan")
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: 'Apple clang version 15.0.0 (clang-1500.3.9.4)'
using SDK: 'MacOSX14.4.sdk'
```

```
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/4.3-arm64/Resources/library/St
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
In file included from /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rc
/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/RcppEigen/include/Eigen/
#include <cmath>
         ^~~~~~
1 error generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000104 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.04 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: 3.933 seconds (Warm-up)
Chain 1:                2.544 seconds (Sampling)
Chain 1:                6.477 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.4 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: 3.956 seconds (Warm-up)
Chain 2:                2.655 seconds (Sampling)
Chain 2:                6.611 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 3.8e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.38 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: 4.247 seconds (Warm-up)
Chain 3:                2.605 seconds (Sampling)
Chain 3:                6.852 seconds (Total)
Chain 3:
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 4e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.4 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: 4.347 seconds (Warm-up)
Chain 4:                2.287 seconds (Sampling)
Chain 4:                6.634 seconds (Total)
Chain 4:
```

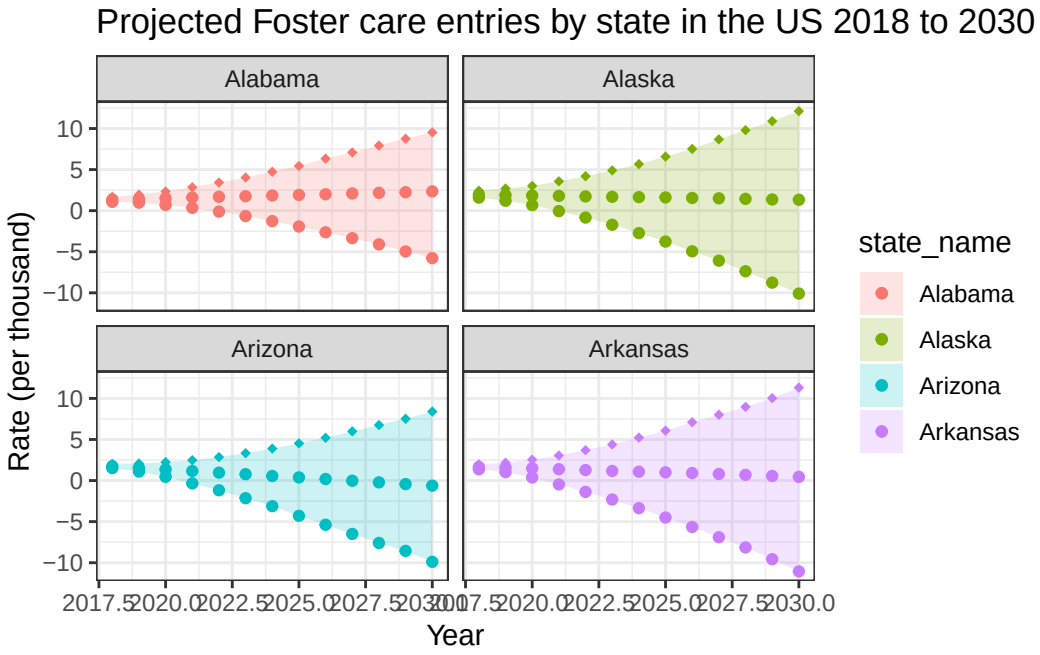
Question 3

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
library(dplyr)
library(tidyr)
library(ggplot2)
library(rstan)

d <- readr::read_csv("fc_entries.csv")

states_of_interest <- c("New York", "Texas", "Florida", "Washington")
d_filtered <- d %>% filter(state %in% states_of_interest)
```

```r
proj_years <- 2018:2030
B.ik_full <- getsplines(c(years, proj_years), 2.5)$B.ik
K <- ncol(B)
K_full <- ncol(B.ik_full)
proj_steps <- K_full - K

alphas <- extract(model)[["alpha"]]
sigmas <- extract(model)[["sigma_alpha"]]
sigma_ys <- extract(model)[["sigma_y"]]
nsims <- nrow(alphas)

states <- unique(d$state)

state_ind <- c(1, 2, 3, 4)

alphas_proj <- array(NA, c(nsims, proj_steps, length(state_ind)))
set.seed(1098)

for(j in state_ind){
  first_next_alpha <- rnorm(n = nsims, mean = 2*alphas[,K,j] - alphas[,K-1,j], sd = sigmas
  second_next_alpha <- rnorm(n = nsims, mean = 2*first_next_alpha - alphas[,K,j], sd = sig

  alphas_proj[,1,j] <- first_next_alpha
  alphas_proj[,2,j] <- second_next_alpha

  for(i in 3:proj_steps){
    alphas_proj[,i,j] <- rnorm(n = nsims,
    mean = 2*alphas_proj[,i-1,j] - alphas_proj[,i-2,j],
    sd = sigmas[,j])
  }
}

y_proj <- array(NA, c(nsims, length(proj_years), length(state_ind)))

for(i in 1:length(proj_years)){
  for(j in state_ind){
    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])
  }
}
```

```r
upper <- function(x) { quantile(x, 0.975) }
lower <- function(x) { quantile(x, 0.025) }

df_all <- c()

for(i in 1:4) {
  state_name <- rep(states[state_ind[i]], length(proj_years))
  ind <- state_ind[i]
  res <- y_proj[,,ind]
  med <- apply(res, 2, median)
  low <- apply(res, 2, lower)
  upp <- apply(res, 2, upper)

  df <- cbind(state_name, proj_years, med, low, upp)
  df_all <- rbind(df_all, df)
}

df_all <- data.frame(df_all)
colnames(df_all) <- c("state_name", "proj_years", "med", "low", "upp")
df_all$proj_years <- as.numeric(as.character(df_all$proj_years))
df_all$med <- as.numeric(as.character(df_all$med))
df_all$low <- as.numeric(as.character(df_all$low))
df_all$upp <- as.numeric(as.character(df_all$upp))

ggplot(df_all, aes(x = proj_years)) +
  geom_point(aes(y = med, color = state_name)) +
  geom_point(aes(y = low, color = state_name)) +
  geom_point(aes(y = upp, color = state_name), pch = 18) +
  geom_ribbon(aes(y = med, ymin = low, ymax = upp, fill = state_name), alpha = 0.2) +
  theme_bw() +
  facet_wrap(~state_name) +
  labs(title = "Projected Foster care entries by state in the US 2018 to 2030",
       x = "Year",
       y = "Rate (per thousand)")
```



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?