Week 11: Splines

29/03/23

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"))
```

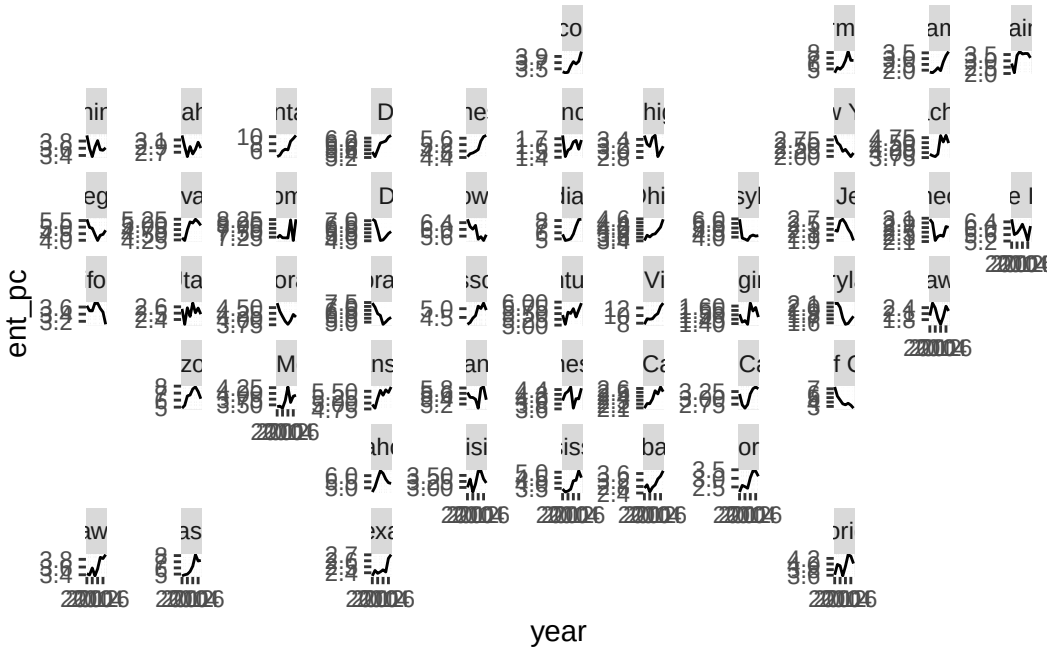
Here's the data

```
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, scales = "free_y")
```



We observed decreasing trend in some states such as New York, New Jersey, and California. But in general, foster care entries per capita increases in most states across years.

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) |>
```

```r
  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)

mod <- stan(data = stan_data, file = "../code/models/lab11.stan")
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEI
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeac
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:9
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'lab11' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000492 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.92 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: 26.9097 seconds (Warm-up)
Chain 1:                17.8018 seconds (Sampling)
Chain 1:                44.7116 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'lab11' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000263 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.63 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: 27.4233 seconds (Warm-up)
Chain 2:                17.9801 seconds (Sampling)
Chain 2:                45.4034 seconds (Total)
Chain 2:
```

```
SAMPLING FOR MODEL 'lab11' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000356 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 3.56 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: 28.4483 seconds (Warm-up)
Chain 3:                19.6366 seconds (Sampling)
Chain 3:                48.0849 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'lab11' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000266 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.66 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: 28.0948 seconds (Warm-up)
Chain 4:                12.1037 seconds (Sampling)
Chain 4:                40.1985 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), I=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 <- extract(mod)[["alpha"]]
sigmas <- extract(mod)[["sigma_alpha"]] # sigma_alpha
sigma_ys <- extract(mod)[["sigma_y"]]
nsims <- nrow(alphas)
```

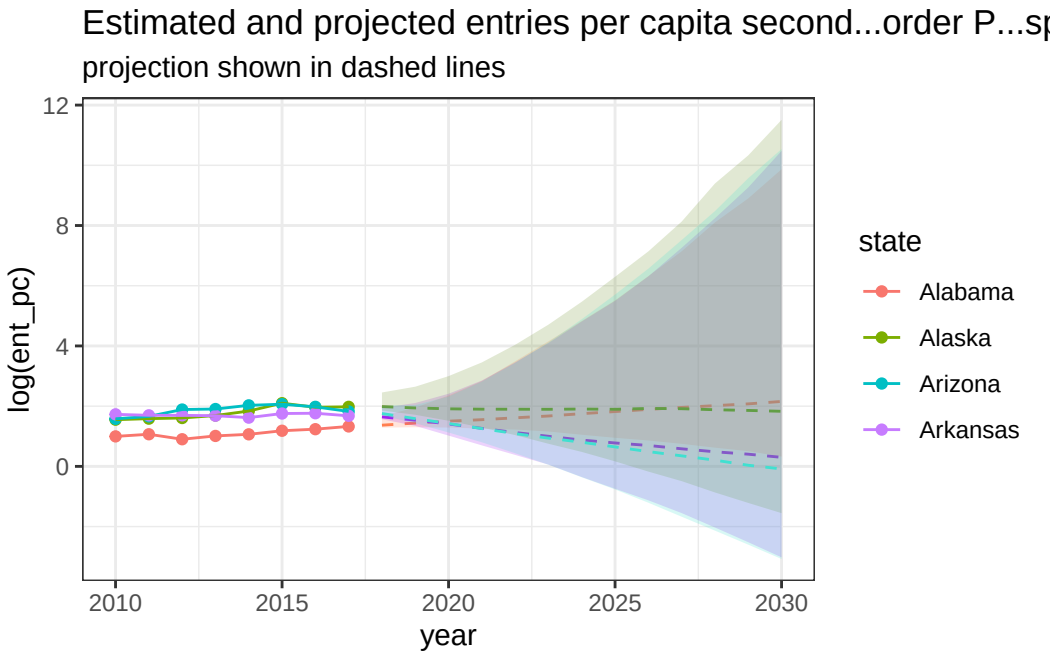
```r
states <- unique(d$state)
# first, project the alphas
alphas_proj <- array(NA, c(nsims, proj_steps, length(states)))
set.seed(1098)
# project the alphas
for(j in 1:length(states)){
  first_next_alpha <- rnorm(n = nsims,
                            mean = 2*alphas[,K,j] - alphas[,K-1,j],
       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

```
# select Alabama, Alaska, Arizona, Arkansas
library(data.table)

Alabama_proj<-transpose(data.frame(y_proj[,,1]))
Alabama_proj$median_ent <- apply(Alabama_proj, 1, median)
Alabama_proj$lower_ent <- apply(Alabama_proj, 1, quantile, probs = c(0.25))
Alabama_proj$upper_ent <- apply(Alabama_proj, 1, quantile, probs = c(0.975))
Alabama_proj$year=2018:2030

Alaska_proj <- transpose(data.frame(y_proj[,,2]))
Alaska_proj$median_ent <- apply(Alaska_proj, 1, median)
Alaska_proj$lower_ent <- apply(Alaska_proj, 1, quantile, probs = c(0.25))
Alaska_proj$upper_ent <- apply(Alaska_proj, 1, quantile, probs = c(0.975))
Alaska_proj$year=2018:2030

Arizona_proj <- transpose(data.frame(y_proj[,,3]))
Arizona_proj$median_ent <- apply(Arizona_proj, 1, median)
Arizona_proj$lower_ent <- apply(Arizona_proj, 1, quantile, probs = c(0.25))
Arizona_proj$upper_ent <- apply(Arizona_proj, 1, quantile, probs = c(0.975))
Arizona_proj$year=2018:2030

Arkansas_proj <- transpose(data.frame(y_proj[,,4]))
Arkansas_proj$median_ent <- apply(Arkansas_proj, 1, median)
Arkansas_proj$lower_ent <- apply(Arkansas_proj, 1, quantile, probs = c(0.25))
```

```r
Arkansas_proj$upper_ent <- apply(Arkansas_proj, 1, quantile, probs = c(0.975))
Arkansas_proj$year=2018:2030
```

```r
d |>
  filter(state %in% c('Alabama', 'Alaska', 'Arizona', 'Arkansas')) |>
  ggplot(aes(year, log(ent_pc)))+
  geom_point(aes(color=state))+
  geom_line(aes(color=state), lty=1)+
  theme_bw()+
  geom_line(data=Alabama_proj, aes(x=year, y=median_ent), linetype="dashed", color="coral"
  geom_ribbon(data = Alabama_proj, aes(x=year, y = median_ent, ymin = lower_ent, ymax = up
    geom_line(data=Arkansas_proj, aes(x=year, y=median_ent), linetype="dashed", color="pur
  geom_ribbon(data = Arkansas_proj, aes(x=year, y = median_ent, ymin = lower_ent, ymax = u
   geom_line(data=Alaska_proj, aes(x=year, y=median_ent), linetype="dashed", color="olived
  geom_ribbon(data = Alaska_proj, aes(x=year, y = median_ent, ymin = lower_ent, ymax = upp
   geom_line(data=Arizona_proj, aes(x=year, y=median_ent), linetype="dashed", color="turqu
  geom_ribbon(data = Arizona_proj, aes(x=year, y = median_ent, ymin = lower_ent, ymax = up
  theme_bw()+
  labs(title = "Estimated and projected entries per capita second-order P-splines", subtit
```



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?