P-splines

Monica Alexander

22/03/2022

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)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(here)
```

```
## here() starts at /cloud/project
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.21.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
##
## Attaching package: 'rstan'
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(tidybayes)
source(here("code/getsplines.R"))
library(geofacet)
library(ggplot2)
```

Here's the data

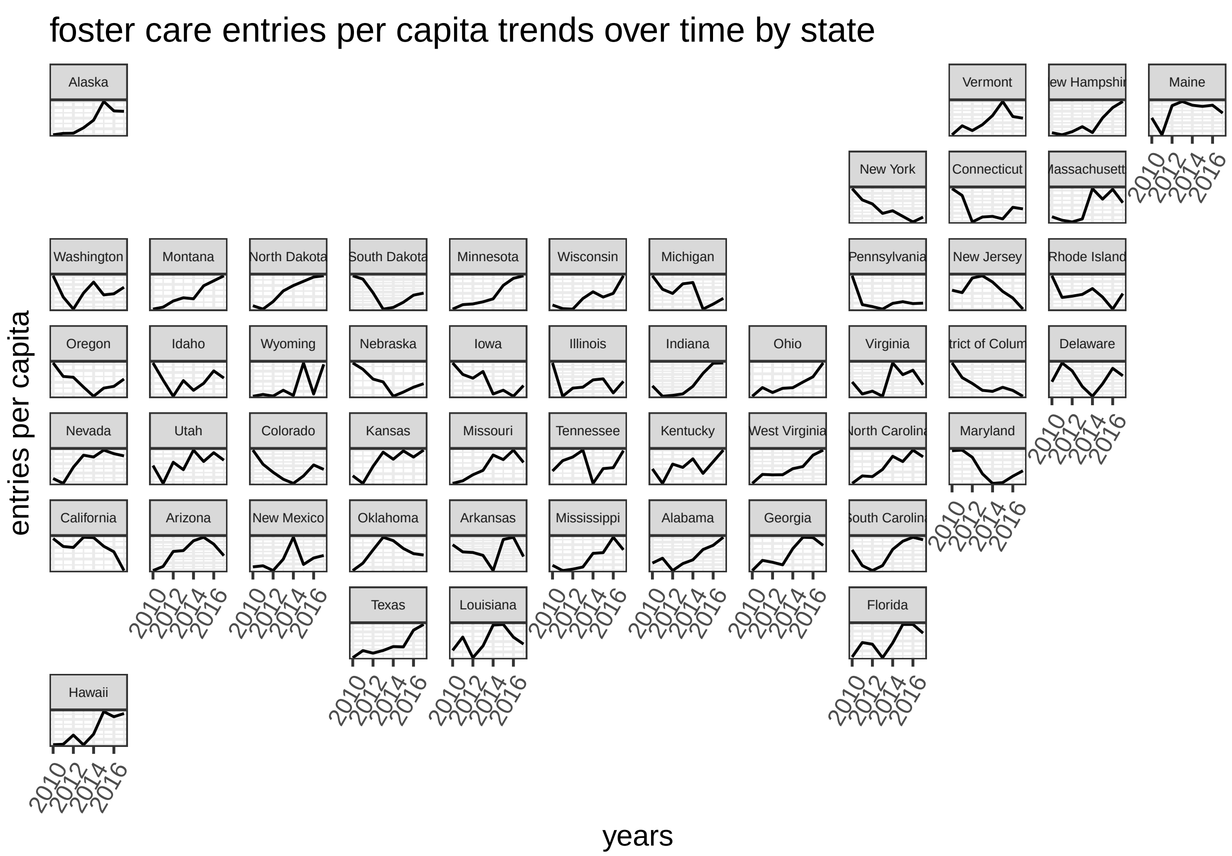
```r
d <- read_csv(here("data/fc_entries.txt"))
```

```
## Rows: 408 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): state
## dbl (5): fips, year, ent, child_acs, ent_pc
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

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
ggplot(d, aes(year, ent, fill = ent)) +
  geom_line() +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 60, hjust = 1),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank()) +
  theme(strip.text.x = element_text(size = 5)) +
  labs(title = "foster care entries per capita trends over time by state",
      x = "years", y = "entries per capita")+
  facet_geo(~ state, grid = "us_state_grid2",scales = "free_y")
```



We see the in many states there's an increasing trend of foster care entries. I suspect that there's a correlation between adjacent states.

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.

```
x.i <- d[1:8,]$ent
I <- 2.5 # between-knot length
res <- getsplines(x.i, I = I) # a function from distortr, to get splines of constant shape
B.ik <- res$B.ik
x.i[1] # look at first value of BMI
```

```
## [1] 3063
```

```
I <- 2.5
res <- getsplines(d$year, I = I)
stan_data <- list(nb = ncol(res$B.ik),
                  ns = length(unique(d$state)),
                  ny = length(unique(d$year)),
                  Y = matrix(log(d$ent_pc), ncol = 51),
                  B = res$B.ik[1:length(unique(d$year)), ])
mod <- stan(data = stan_data,
            file = "spline.stan",
            iter = 5000,
            seed = 2)
```

```
## Trying to compile a simple C file
```

```
## Running /opt/R/4.1.2/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/opt/R/4.1.2/lib/R/include" -DNDEBUG   -I"/cloud/lib/x86_64-pc-linux-gnu-library/4.1/Rcpp/incl
## In file included from /cloud/lib/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:88,
##                  from /cloud/lib/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##                  from /cloud/lib/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prim/m
##                  from <command-line>:
## /cloud/lib/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: erro
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /cloud/lib/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: err
##   628 | namespace Eigen {
##       |                 ^
## In file included from /cloud/lib/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##                  from /cloud/lib/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prim/m
##                  from <command-line>:
## /cloud/lib/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:96:10: fatal error: complex:
##    96 | #include <complex>
```

```
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/opt/R/4.1.2/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL 'spline' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000147 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.47 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 16.7124 seconds (Warm-up)
## Chain 1:                13.1822 seconds (Sampling)
## Chain 1:                29.8945 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'spline' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 9.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.98 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 17.23 seconds (Warm-up)
## Chain 2:                13.1847 seconds (Sampling)
## Chain 2:                30.4147 seconds (Total)
## Chain 2:
##
```

```
## SAMPLING FOR MODEL 'spline' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 9.7e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.97 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 16.7551 seconds (Warm-up)
## Chain 3:                25.6455 seconds (Sampling)
## Chain 3:                42.4006 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'spline' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9.7e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.97 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 18.505 seconds (Warm-up)
## Chain 4:                13.0267 seconds (Sampling)
## Chain 4:                31.5318 seconds (Total)
## Chain 4:
```

```
## Warning: There were 168 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing Information was low. Se
```

```
## https://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: The largest R-hat is 1.07, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

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.

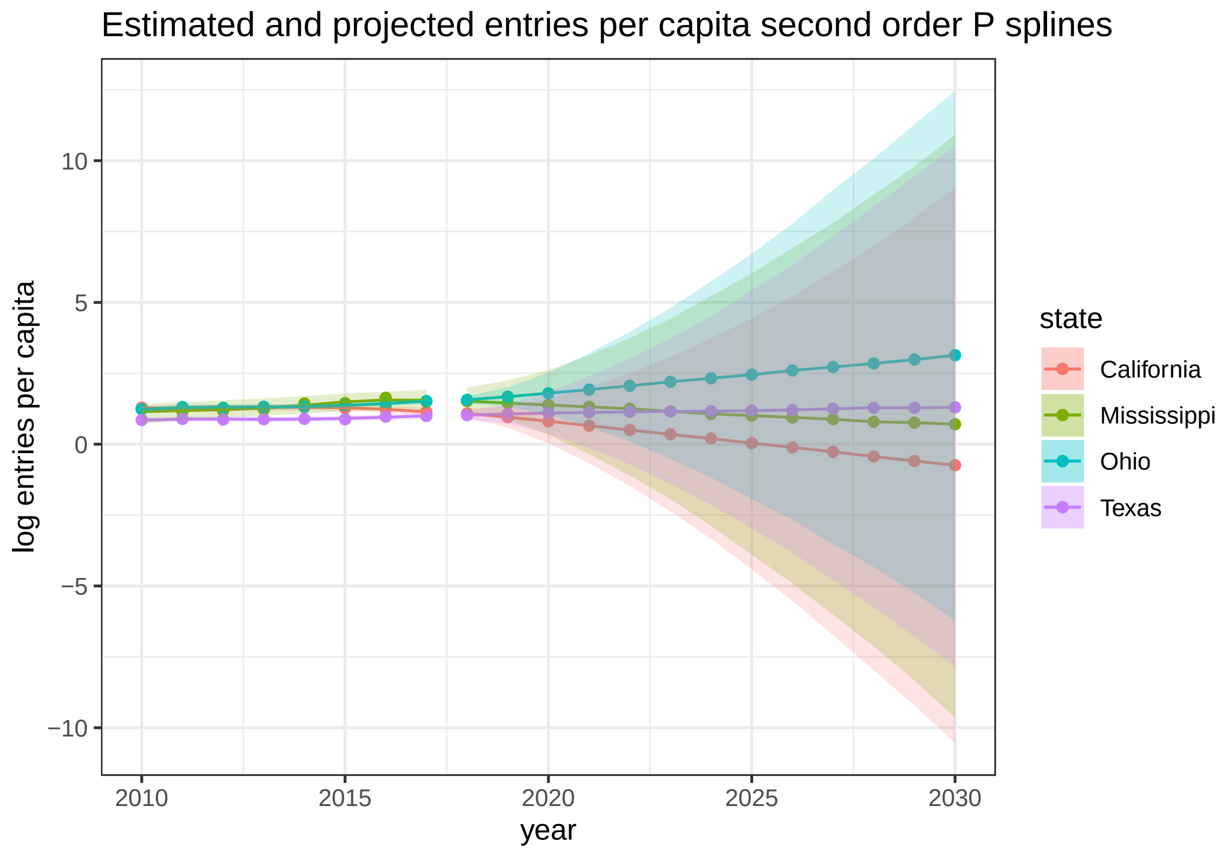
```r
years <- 2010:2017
proj_years <- 2018:2030
B.ik_full <- getsplines(c(2010:2017, proj_years), I = 2.5)$B.ik
K <- ncol(res$B.ik)
K_full <- ncol(B.ik_full)
proj_steps <- K_full - K
alphas <- extract(mod)[["alpha"]]
sigmas <- exp(extract(mod)[["log_sigma_alpha"]])
sigma_ys <- extract(mod)[["sigma"]]
nsims <- nrow(alphas)
states <- unique(d$state)
alphas_proj <- array(NA, c(nsims, proj_steps, length(states)))
set.seed(1098)
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
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(states)))
for(i in 1:length(proj_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])
    }
}
```

```r
res1 <- mod %>%
  gather_draws(Y_rep[condition1, condition2]) %>%
  median_qi()
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## Please use `gather()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```r
colnames(res1)[1:2] <- c("year", "state")
index <- which(states %in% c("California", "Mississippi", "Ohio", "Texas"))
res2 <- matrix(NA, nrow = 663, ncol = 5)
res2[, 1] <- rep(1:length(proj_years), each = 51)
res2[, 2] <- rep(1:51, length(proj_years))
colnames(res2) <- c("year", "state", ".value", ".lower", ".upper")
res2 <- data.frame(res2)
for (i in 1:663) {
  res2[i, ".value"] <- median(y_proj[, res2$year[i], res2$state[i]])
  res2[i, ".lower"] <- quantile(y_proj[, res2$year[i], res2$state[i]], 0.025)
  res2[i, ".upper"] <- quantile(y_proj[, res2$year[i], res2$state[i]], 0.975)
}
res1_plot <- res1 %>%
  filter(state %in% index)
res1_plot$year <- rep(years, each = 4)
res1_plot$state <- rep(c("California", "Mississippi", "Ohio", "Texas"),
                       length(unique(res1_plot$year)))
res2_plot <- res2 %>%
  filter(state %in% index)
res2_plot$year <- rep(proj_years, each = 4)
res2_plot$state <- rep(c("California", "Mississippi", "Ohio", "Texas"),
                       length(unique(res2_plot$year)))
d_plot <- d %>%
  filter(state %in% c("California", "Mississippi", "Ohio", "Texas"))
d_plot$log_ent_pc <- log(d_plot$ent_pc)
ggplot(d_plot, aes(year, log_ent_pc)) +
  geom_point(aes(color = state)) +
  geom_line(data = res1_plot, aes(year, .value, col = state)) +
  geom_ribbon(data = res1_plot, aes(x = year, y = .value, ymin = .lower, ymax = .upper, fill =
                                      state), alpha = 0.2) +
  theme_bw() +
  geom_line(data = res2_plot, aes(year, .value, col=state)) +
  geom_point(data = res2_plot, aes(year, .value, col=state)) +
  geom_ribbon(data = res2_plot, aes(y = .value, ymin = .lower, ymax = .upper, fill=state), alpha
              = 0.2) +
  theme_bw() +
  labs(title = "Estimated and projected entries per capita second order P splines",
       y = "log entries per capita")
```


Estimated and projected entries per capita second order P splines
log entries per capita
year
10
5
0
−5
−10
2010
2015
2020
2025
2030
state
California
Mississippi
Ohio
Texas