Week 11: Splines

Alice Huang

today

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 1.3.2 --
## v ggplot2 3.4.0     v purrr   0.3.5
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.2.1     v stringr 1.4.1
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(here)
```

```
## here() starts at C:/Users/Alice/Desktop/GRAD SCHOOL/STA2201/STA2201AppliedStatAliceHuang
```

```
library(rstan)
```

```
## Loading required package: StanHeaders
##
## rstan version 2.26.13 (Stan version 2.26.1)
##
## 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)
## For within-chain threading using ‘reduce_sum()‘ or ‘map_rect()‘ Stan functions,
## change ‘threads_per_chain‘ option:
## rstan_options(threads_per_chain = 1)
##
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
##
## Attaching package: 'rstan'
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(tidybayes)
source(here("getsplines.R"))
```

Here's the data

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

```
## 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.
```

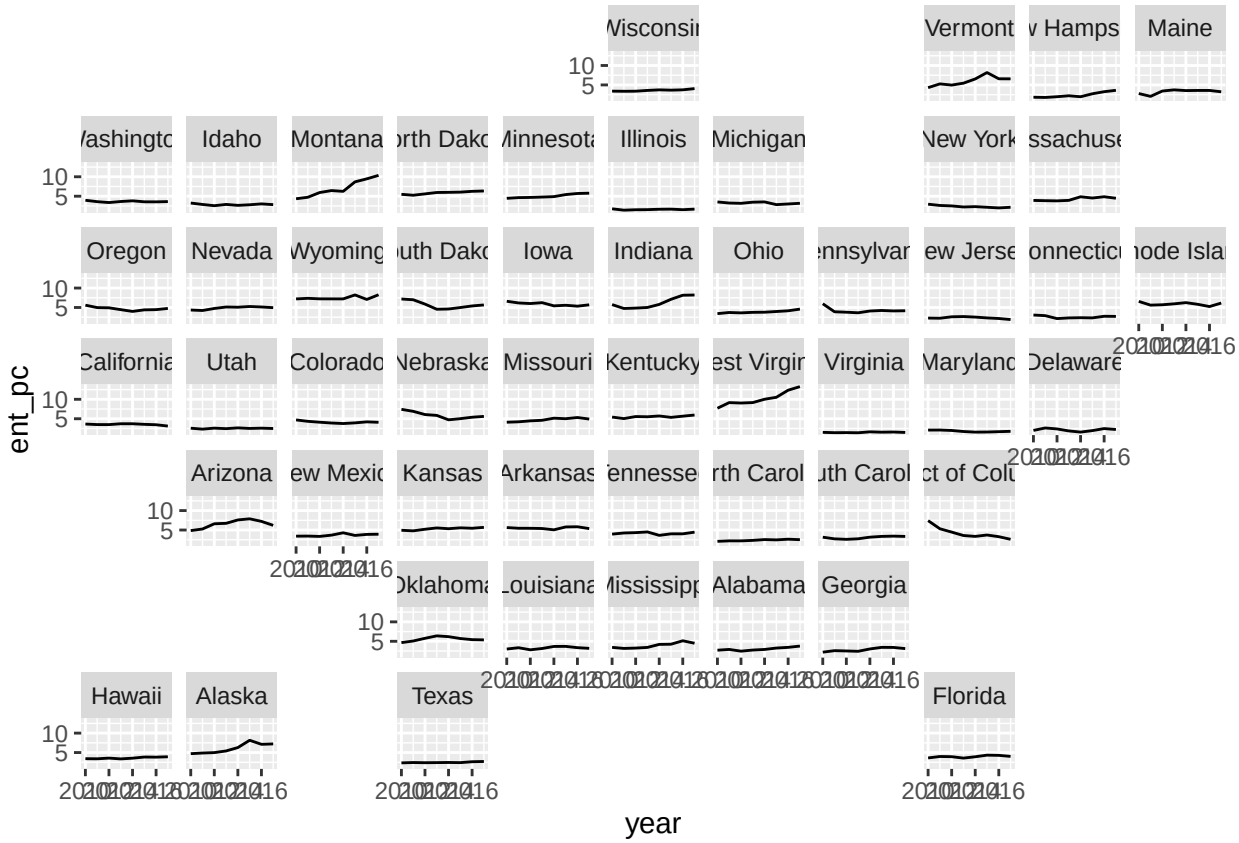
Question 1

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.

```
library(geofacet)
```

```
## Warning: package 'geofacet' was built under R version 4.2.3
```

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



For most states, foster care entries seemed to have stayed around the same from 2010 - 2016. However for Montana, West Virginia, Alaska, Vermont, there were increases in foster care entries. For the District of Columbia there was a decrease in foster care entries.

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())

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 = "lab11.stan")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000409 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 4.09 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: 263.44 seconds (Warm-up)
## Chain 1:                255.007 seconds (Sampling)
## Chain 1:                518.447 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000237 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.37 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: 243.177 seconds (Warm-up)
## Chain 2:                246.401 seconds (Sampling)
## Chain 2:                489.578 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000245 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.45 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: 240.295 seconds (Warm-up)
## Chain 3:                246.615 seconds (Sampling)
## Chain 3:                486.91 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000276 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.76 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: 241.805 seconds (Warm-up)
```

```
## Chain 4:                244.795 seconds (Sampling)
## Chain 4:                486.6 seconds (Total)
## Chain 4:
```

```
## Warning: There were 4000 transitions after warmup that exceeded the maximum treedepth. Increase max_
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

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

```
## Warning: The largest R-hat is 4.42, 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 quanti
## 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.

```
years = 2010:2017
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))$B.ik
```

```
## Error in getsplines(c(years, proj_years)): argument "I" is missing, with no default
```

```
K <- ncol(B.ik) # number of knots in sample
```

```
## Error in ncol(B.ik): object 'B.ik' not found
```

```
K_full <- ncol(B.ik_full) # number of knots over entire period
```

```
## Error in ncol(B.ik_full): object 'B.ik_full' not found
```

```
proj_steps <- K_full - K # number of projection steps
```

```
## Error in eval(expr, envir, enclos): object 'K_full' not found
```

```
# get your posterior samples
alphas <- extract(mod)[["alpha"]]
sigmas <- extract(mod)[["sigma"]] # sigma_alpha
sigma_ys <- extract(mod)[["sigma_y"]]
nsims <- nrow(alphas)

states = c("California", "Mississippi", "Ohio", "Texas")

# first, project the alphas
alphas_proj <- array(NA, c(nsims, proj_steps, length(states)))
```

```
## Error in array(NA, c(nsims, proj_steps, length(states))): object 'proj_steps' not found
```

```
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])
}
}
```

```
## Error in rnorm(n = nsims, mean = 2 * alphas[, K, j] - alphas[, K - 1, : invalid arguments
```

```
# 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])
}
}
```

```
## Error in cbind(alphas[, , j], alphas_proj[, , j]): object 'alphas_proj' not found
```

```
# then proceed as normal to get median, quantiles etc
```

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?

Model might not know what part of the model is due to splines or covariates. Need the mean in the deviations to be 0.