Lab 11 Submission

Yeonjoon Choi

2023-04-01

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 3.4.0     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
## Warning: package 'tibble' was built under R version 4.2.2
```

```
## Warning: package 'dplyr' was built under R version 4.2.2
```

```
## Warning: package 'forcats' was built under R version 4.2.2
```

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

```
library(here)
```

```
## Warning: package 'here' was built under R version 4.2.2
```

```
## here() starts at C:/Users/choip/OneDrive/Documents/GitHub/STA2201H_Yeonjoon_Choi/Lab 11 submission
```

```
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("getsplines.R")
```

Here's the data

```
d <- read_csv("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

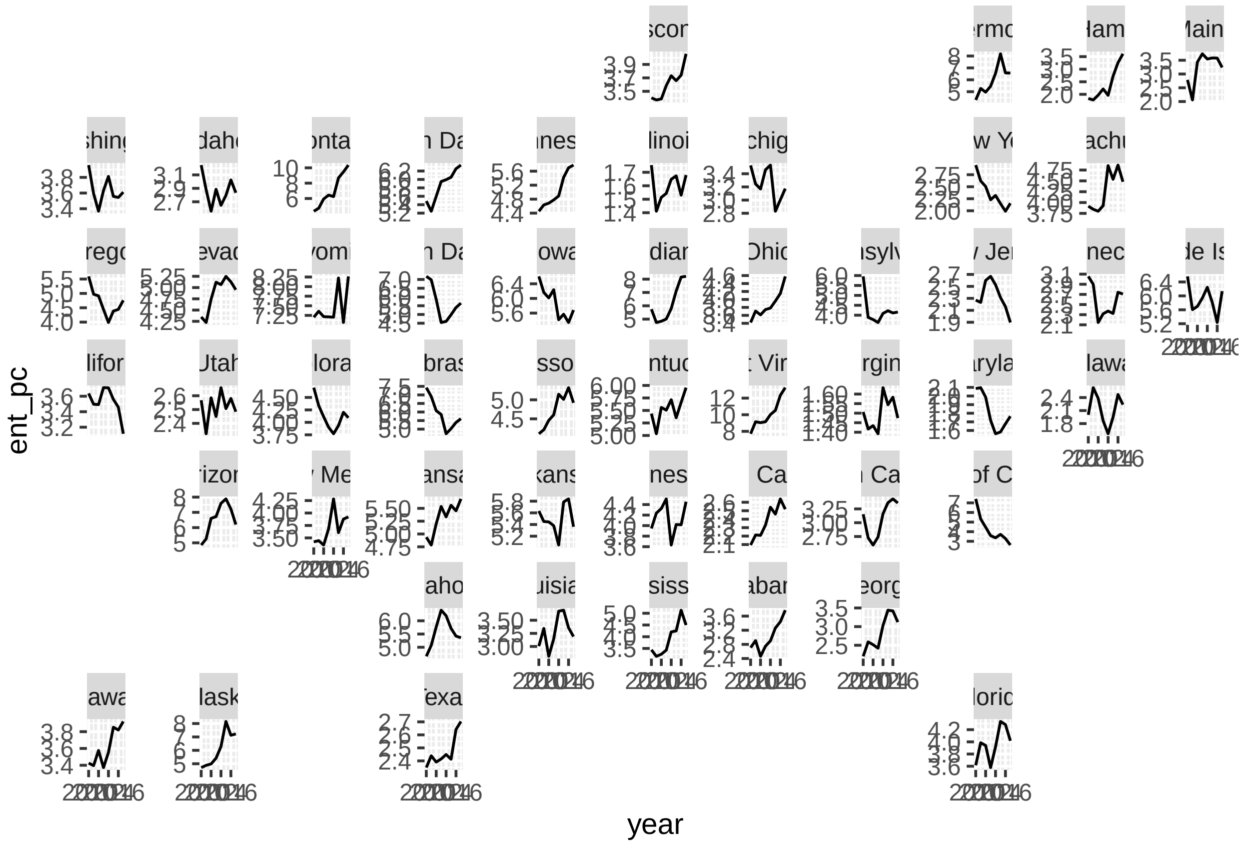
Make a plot highlighting trends over time by state. Might be a good opportunity to use `geofacet`. Describe what you see in a couple of sentences.

```
require(geofacet)
```

```
## Loading required package: geofacet
```

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

```
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$year)
N <- length(years)
y <- log(d |>
  select(state, year, ent_pc) |>
  pivot_wider(names_from = "state", values_from = "ent_pc") |>
  select(-year) |>
  as.matrix())
res <- getsplines(years, 2.5)
B.ik <- res$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 = "lab11.stan")



#saveRDS(mod, file = "lab11_stan.RDS")

mod = readRDS("lab11_stan.RDS")
```

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.

```
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)

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] )
```

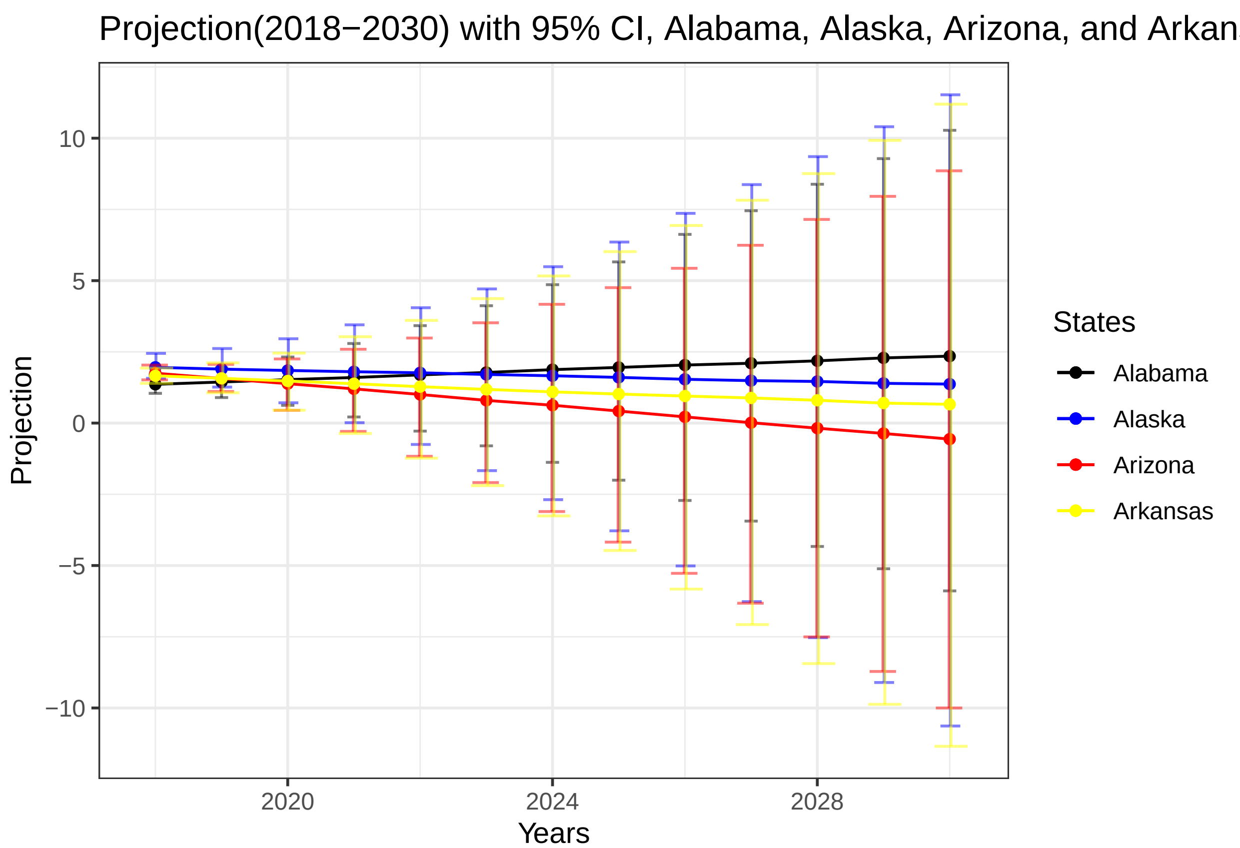
```r
this_lambda <- all_alphas %*% as.matrix(B.ik_full[length(years)+i, ])
y_proj[,i,j] <- rnorm(n = nsims, mean = this_lambda, sd = sigma_ys[,j])
}
}
# then proceed as normal to get median, quantiles etc
```

We choose Alabama, Alaska, Arizona, and Arkansas.

```r
AL = list()
for (i in 1:4000){
  AL[[i]] = y_proj[i, ,1]
}

AL = do.call(rbind,AL)

AL = as.data.frame(AL)

results_med = sapply(AL,  median)

get_up = function(x){
  return (quantile(x, 0.975))
}

get_down = function(x){
  return (quantile(x, 0.025))
}

results_up = sapply(AL, get_up)
results_down = sapply(AL, get_down)

df = as.data.frame(cbind(cbind(results_med, results_up), results_down))


ALA = list()
for (i in 1:4000){
  ALA[[i]] = y_proj[i, ,2]
}

ALA = do.call(rbind,ALA)

ALA = as.data.frame(ALA)

results_med_1 = sapply(ALA,  median)

get_up = function(x){
  return (quantile(x, 0.975))
}

get_down = function(x){
  return (quantile(x, 0.025))
}

results_up_1 = sapply(ALA, get_up)
```

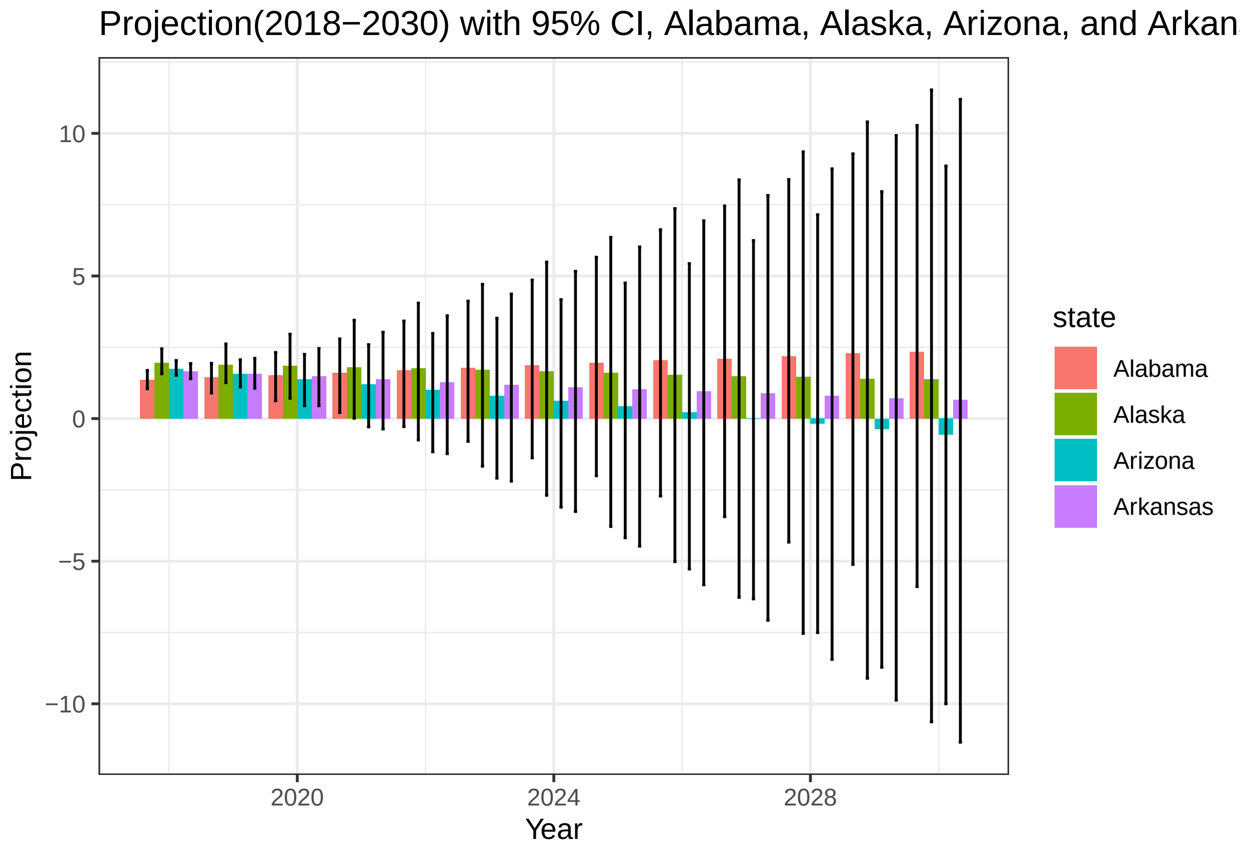
```r
results_down_1 = sapply(ALA, get_down)



df_1 = as.data.frame(cbind(cbind(results_med_1, results_up_1), results_down_1))



AZ = list()
for (i in 1:4000){
  AZ[[i]] = y_proj[i, ,3]
}

AZ = do.call(rbind,AZ)

AZ = as.data.frame(AZ)

results_med_2 = sapply(AZ,  median)

get_up = function(x){
  return (quantile(x, 0.975))
}

get_down = function(x){
  return (quantile(x, 0.025))
}

results_up_2 = sapply(AZ, get_up)
results_down_2 = sapply(AZ, get_down)



df_2 = as.data.frame(cbind(cbind(results_med_2, results_up_2), results_down_2))

AK = list()
for (i in 1:4000){
  AK[[i]] = y_proj[i, ,4]
}

AK = do.call(rbind,AK)

AK = as.data.frame(AK)

results_med_3 = sapply(AK,  median)

get_up = function(x){
  return (quantile(x, 0.975))
}

get_down = function(x){
  return (quantile(x, 0.025))
}
```

```
results_up_3 = sapply(AK, get_up)
results_down_3 = sapply(AK, get_down)



df_3 = as.data.frame(cbind(cbind(results_med_3, results_up_3), results_down_3))


ggplot() +
  geom_line(data = df, aes(x = 2018:2030, y = results_med, col = I("black"))) +
  geom_point(data = df, aes(x = 2018:2030, y = results_med, col = I("black")))+
  geom_errorbar(data = df, aes(x = 2018:2030,ymin = results_down, ymax = results_up), width = 0.2, col =
  geom_line(data = df_1, aes(x = 2018:2030, y = results_med_1,   col = I("blue"))) +
  geom_point(data = df_1, aes(x = 2018:2030, y = results_med_1, col = I("blue")))+
  geom_errorbar(data = df_1, aes(x = 2018:2030+0.01, ymin = results_down_1, ymax = results_up_1), width
  geom_line(data = df_2, aes(x = 2018:2030, y = results_med_2, col = I("red"))) +
  geom_point(data = df_2, aes(x = 2018:2030, y = results_med_2, col = I("red")))+
  geom_errorbar(data = df_2, aes(x = 2018:2030-0.01,ymin = results_down_2, ymax = results_up_2), width =
  geom_line(data = df_3, aes(x = 2018:2030, y = results_med_3, col = I("yellow"))) +
  geom_point(data = df_3, aes(x = 2018:2030, y = results_med_3, col = I("yellow")))+
  geom_errorbar(data = df_3, aes(x = 2018:2030+0.02,ymin = results_down_3, ymax = results_up_3), width =
  labs(x = "Years", y = "Projection", title = "Projection(2018-2030) with 95% CI, Alabama, Alaska, Arizo
   scale_color_manual(values=c("black", "blue", "red", "yellow"),
                     labels=c("Alabama", "Alaska", "Arizona", "Arkansas"),
                     name = "States") +
  theme_bw()
```



The plot is hard to read. Hence, we also add a barplot.

```
colnames(df_1) = colnames(df)
colnames(df_2) = colnames(df)
colnames(df_3) = colnames(df)

df$state = rep("Alabama", nrow(df))
df_1$state = rep("Alaska", nrow(df))
df_2$state = rep("Arizona", nrow(df))
df_3$state = rep("Arkansas", nrow(df))

df$year  = 2018:2030
df_1$year  = 2018:2030
df_2$year  = 2018:2030
df_3$year  = 2018:2030

data = as.data.frame(rbind(rbind(rbind(df, df_1), df_2),df_3))

ggplot(data, aes(x = year, y = results_med, fill = state))+
  geom_bar(position = "dodge", stat="identity")+
  geom_errorbar(aes(ymin = results_down, ymax = results_up), width = 0.2,
              position = position_dodge(width = 0.9))+
  ylab("Projection")+
  xlab("Year")+
  labs(title = "Projection(2018-2030) with 95% CI, Alabama, Alaska, Arizona, and Arkansas")+
  theme_bw()
```



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?

To avoid identifiability issue, we need to constrain so that splines sum to 0.