Lab Week 11

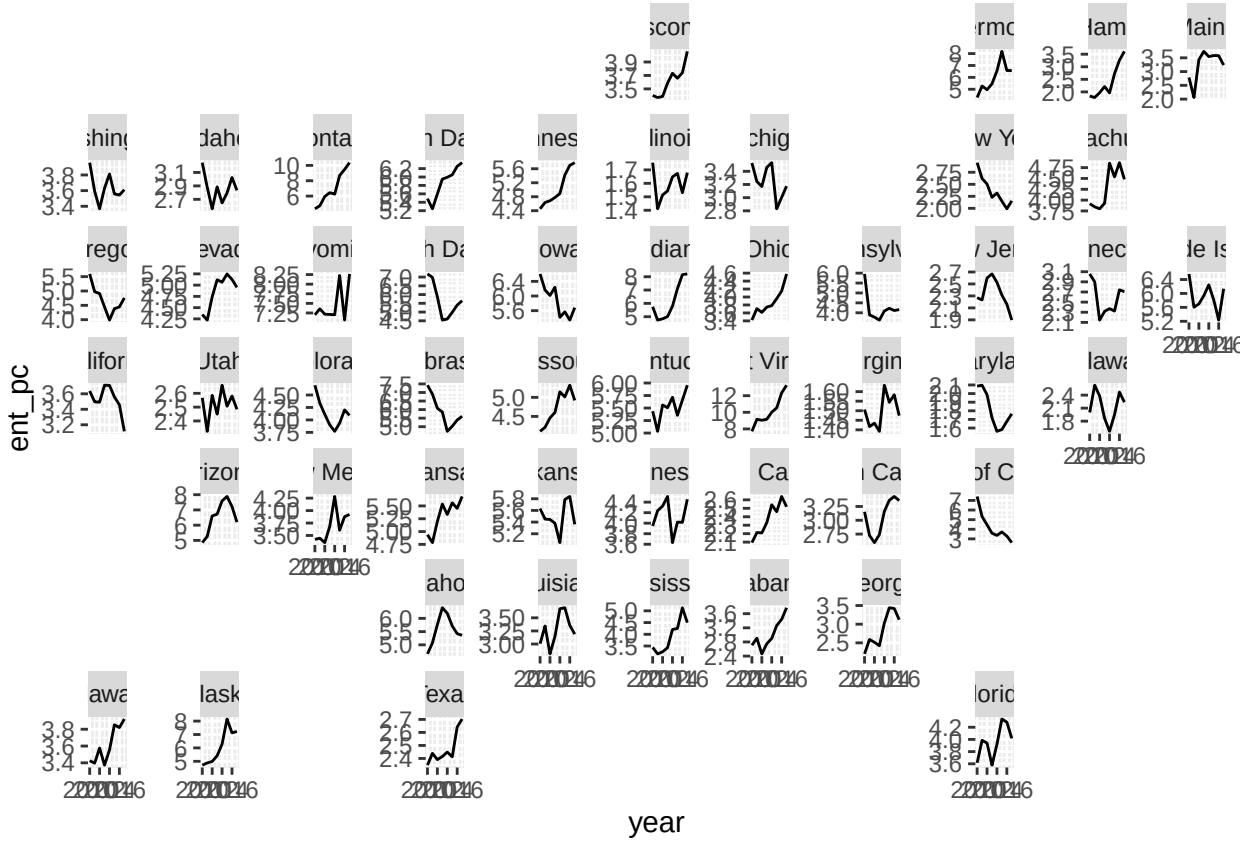
Rosa Fallahpour

```
d <- read.csv("https://raw.githubusercontent.com/MJAlexander/applied-stats-2023/main/data/fc_entries.csv
```

Question 1

The following graph illustrates varying trends across different states. In some states, we observe an increasing trend, while others show a decreasing trend. Additionally, certain states display no general pattern, exhibiting fluctuations instead.

```
library(geofacet)
library(ggplot2)
d |>
  ggplot(aes(year, ent_pc)) +
  geom_line()+
  facet_geo(~state, scales = "free_y")
```



Question 2

The model we want to fit is as below. Where $y_{s,t}$ is the logged entries per capita for state s in year t.

$$
\begin{aligned}
y_{st} &\sim N(\log \lambda_{st}, \sigma^2_{y,s}) \\
\log \lambda_{st} &= \alpha_k B_k(t) \\
\Delta^2 \alpha_k &\sim N(0, \sigma^2_{\alpha,s}) \\
\log \sigma_{\alpha,s} &\sim N(\mu_\sigma, \tau^2)
\end{aligned}
$$

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

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000254 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.54 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: 11.752 seconds (Warm-up)
## Chain 1:                7.8 seconds (Sampling)
## Chain 1:                19.552 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.00012 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.2 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: 12.39 seconds (Warm-up)
## Chain 2:                8.641 seconds (Sampling)
## Chain 2:                21.031 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000109 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 1.09 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: 11.784 seconds (Warm-up)
## Chain 3:                8.574 seconds (Sampling)
## Chain 3:                20.358 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000115 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 1.15 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: 12.349 seconds (Warm-up)
## Chain 4:                7.632 seconds (Sampling)
## Chain 4:                19.981 seconds (Total)
## Chain 4:
```

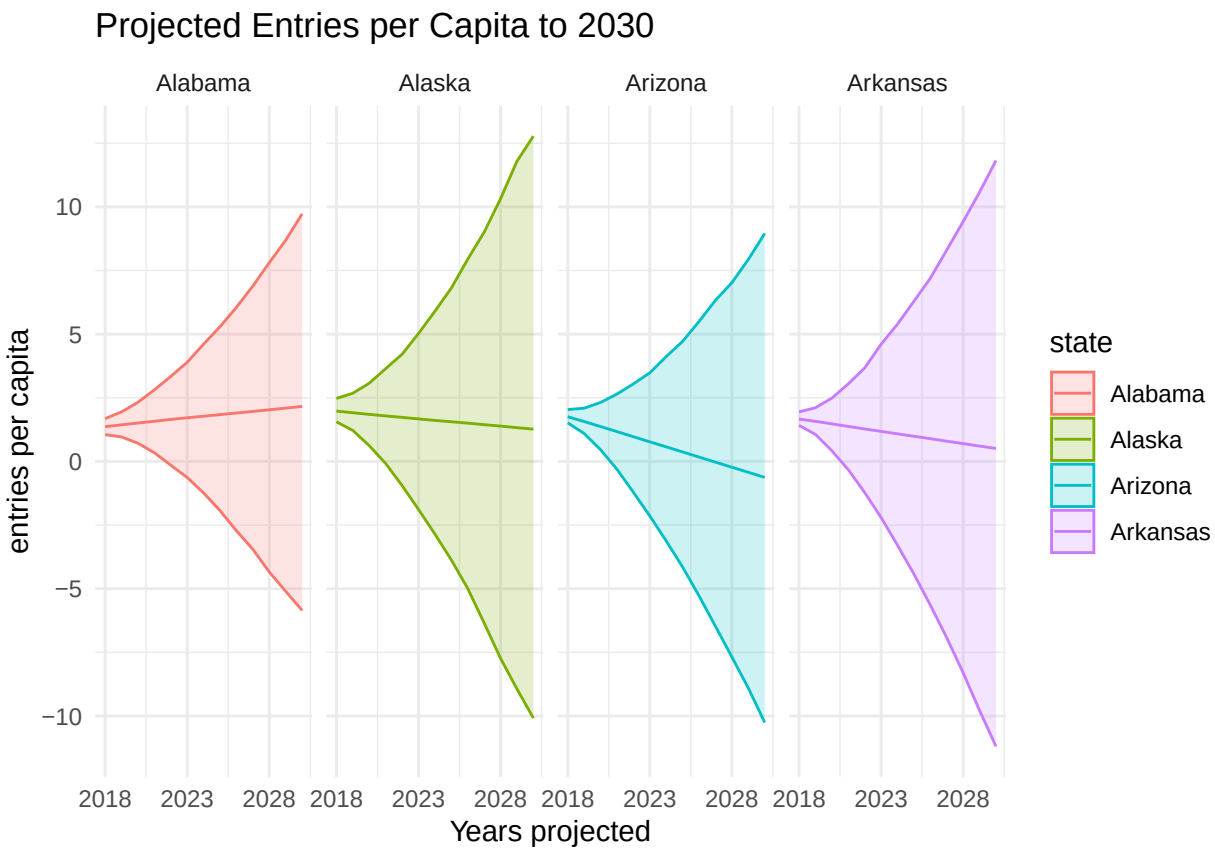
Question 3

The plots below display the projected entries per capita up to 2030 for four selected states:

```
states <- unique(d$state)
proj_years <- 2018:2030
B.ik_full <- getsplines(c(years, proj_years), 2.5)$B.ik
K <- ncol(B) # number of knots in sample
K_full <- ncol(B.ik_full) # number of knots over entire
# get your posterior samples
alphas <- extract(mod)[["alpha"]]
sigmas <- extract(mod)[["sigma_alpha"]] # sigma_alpha
sigma_ys <- extract(mod)[["sigma_y"]]
nsims <- nrow(alphas)
proj_steps <- K_full - K # number of projection steps
# 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])
}
}
```

```
states_s <- c(1, 2, 3, 4)
y_proj_s <- y_proj[,,states_s]
y_proj_mean <- apply(y_proj_s, c(2, 3), mean)
y_proj_lower <- apply(y_proj_s, c(2, 3), function(x) quantile(x, 0.025))
y_proj_upper <- apply(y_proj_s, c(2, 3), function(x) quantile(x, 0.975))
state_names <- unique(d$state)[states_s]
df <- data.frame(
  year = rep(proj_years, length(states_s)),
  mean = as.vector(y_proj_mean),
  lower = as.vector(y_proj_lower),
  upper = as.vector(y_proj_upper),
  state = rep(state_names, each = length(proj_years))
)
ggplot(df, aes(x = year, y = mean, group = state, color = state)) +
  geom_line() +
  geom_ribbon(aes(ymin = lower, ymax = upper, fill = state), alpha = 0.2) +
  labs(title = "Projected Entries per Capita to 2030",
       x = "Years projected",
       y = "entries per capita") +
  theme_minimal() +
  facet_wrap(~state, ncol = length(states_s)) +
  scale_x_continuous(breaks = seq(min(proj_years), max(proj_years), by = 5))
```


Projected Entries per Capita to 2030
Alabama
Alaska
Arizona
Arkansas
entries per capita
10
5
0
−5
−10
2018
2023
2028
Years projected
state
Alabama
Alaska
Arizona
Arkansas