Week 11: Splines

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.

Here's the data

```
# d <- read_csv(here('data/fc_entries.csv'))
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

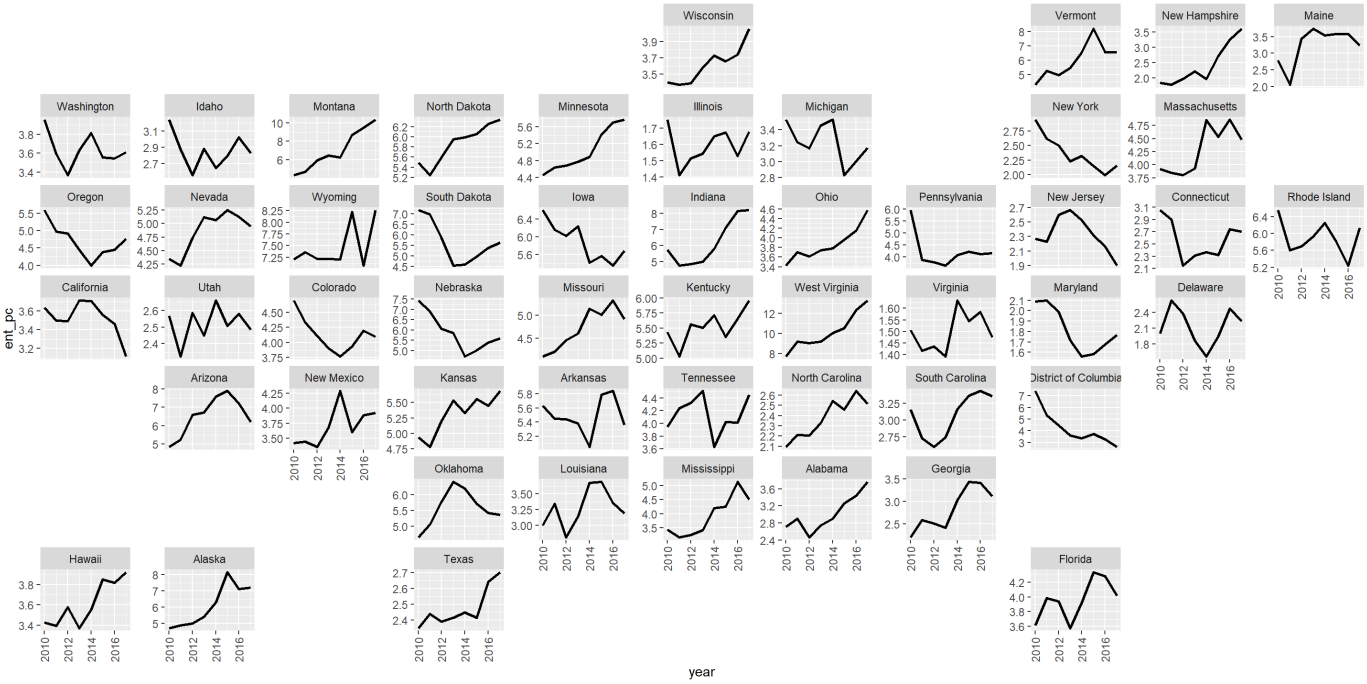
Make a plot highlighting trends over time by state. Might be a good opportunity to use `geofacet`. Describe what you see in a couple of sentences.

```
library(geofacet)
```

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

```
d |>
    ggplot(aes(year, ent_pc)) + geom_line(size = 1) + facet_geo(~state, scales = "free_y") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
```



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 = "./spline_mod.stan")
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.00096 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 9.6 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: 45.361 seconds (Warm-up)
## Chain 1:                21.178 seconds (Sampling)
## Chain 1:                66.539 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000358 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 3.58 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: 31.909 seconds (Warm-up)
## Chain 2:                18.69 seconds (Sampling)
## Chain 2:                50.599 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000219 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.19 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: 31.647 seconds (Warm-up)
## Chain 3:                18.896 seconds (Sampling)
## Chain 3:                50.543 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.00022 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.2 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: 29.62 seconds (Warm-up)
## Chain 4:                18.469 seconds (Sampling)
## Chain 4:                48.089 seconds (Total)
## Chain 4:
```

```
## Warning: There were 6 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: Examine the pairs() plot to diagnose sampling problems
```

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.

```
# 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_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
```

```
states <- unique(d$state)
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])
    }
}
```

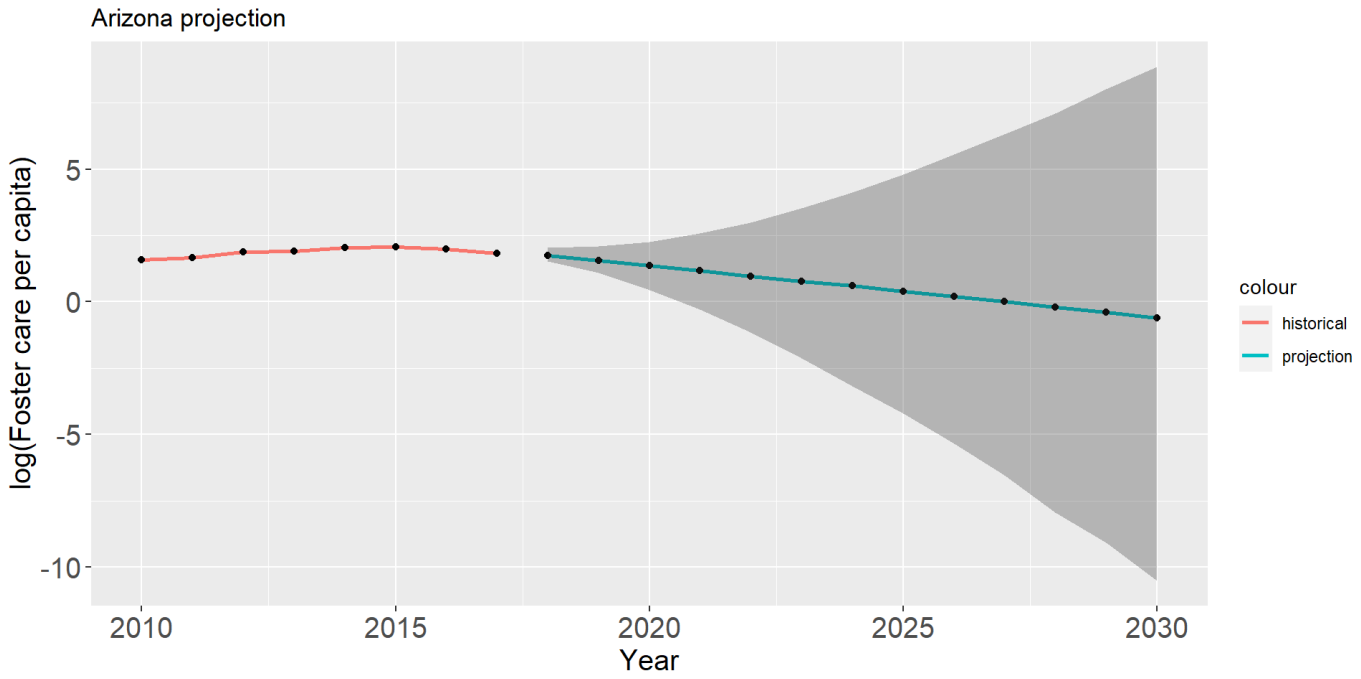
```
# now use these to get y's
y_proj <- array(NA, c(nsims, length(proj_years), length(states)))
for (i in 1:length(proj_years)) {
    # now over years
    for (j in 1:length(states)) {
        all_alphas <- cbind(alphas[, , j], alphas_proj[, , j])
        this_lambda <- all_alphas %*% as.matrix(B.ik_full[length(years) + i, ])
        y_proj[, i, j] <- rnorm(n = nsims, mean = this_lambda, sd = sigma_ys[, j])
    }
}
```

Plotting the results

```
# Utility function to calculate the summary statistics
calc_dstats <- function(state_data) {
    # nas <- array(NA, length(proj_years))
    dstats <- tibble(mean = apply(state_data, 2, mean), median = apply(state_data,
        2, median), lb = apply(state_data, 2, quantile, probs = 0.025), ub = apply(state_dat
a,
        2, quantile, probs = 0.975), years = proj_years)
    return(dstats)
}
```

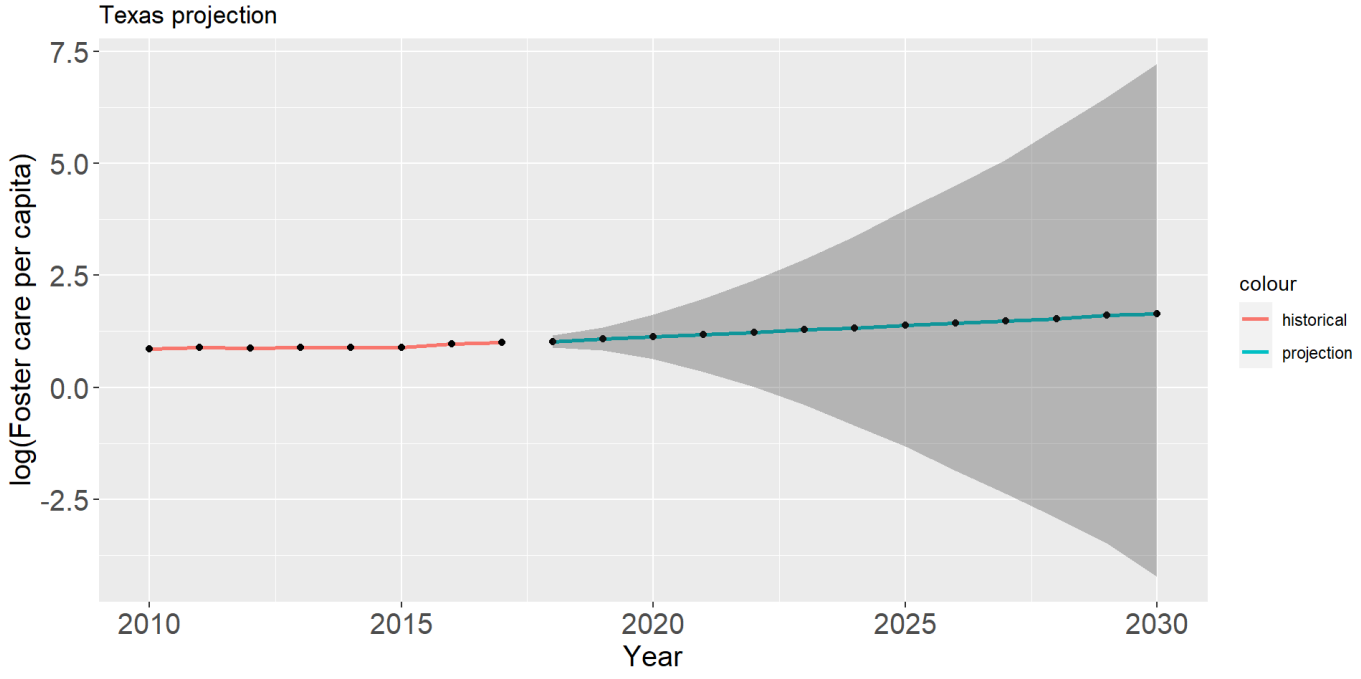
Projections for the state of Arizona

```
state_name = "Arizona"
state_id = which(states == state_name)
arizona <- calc_dstats(y_proj[, , state_id])
arizona_historic <- d %>%
    filter(state == state_name)

ggplot(data = arizona) + geom_line(aes(x = years, y = median, color = "projection"),
    linewidth = 1) + geom_point(aes(x = years, y = median)) + geom_ribbon(mapping = aes(x = y
ears,
    ymin = lb, ymax = ub), alpha = 0.3) + ggtitle("Arizona projection") + geom_line(data = ar
izona_historic,
    aes(x = years, y = log(ent_pc), color = "historical"), linewidth = 1) + geom_point(data =
arizona_historic,
    aes(x = years, y = log(ent_pc))) + theme_large_text + labs(x = "Year", y = "log(Foster ca
re per capita)")
```



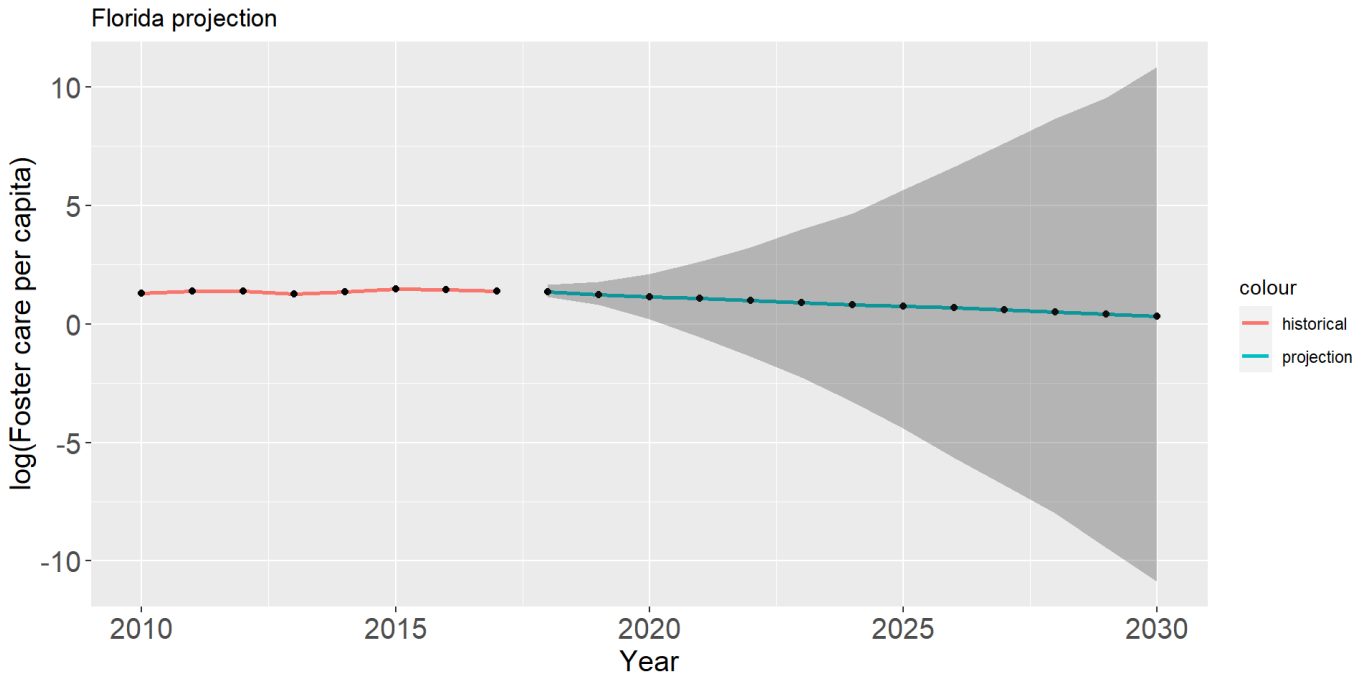
Projections for the state of Texas

```
state_name = "Texas"
state_id = which(states == state_name)
texas <- calc_dstats(y_proj[, , state_id])
texas_historic <- d %>%
    filter(state == state_name)

ggplot(data = texas) + geom_line(aes(x = years, y = median, color = "projection"),
    linewidth = 1) + geom_point(aes(x = years, y = median)) + geom_ribbon(mapping = aes(x = y
ears,
    ymin = lb, ymax = ub), alpha = 0.3) + ggtitle("Texas projection") + geom_line(data = texa
s_historic,
    aes(x = years, y = log(ent_pc), color = "historical"), linewidth = 1) + geom_point(data =
texas_historic,
    aes(x = years, y = log(ent_pc))) + theme_large_text + labs(x = "Year", y = "log(Foster ca
re per capita)")
```



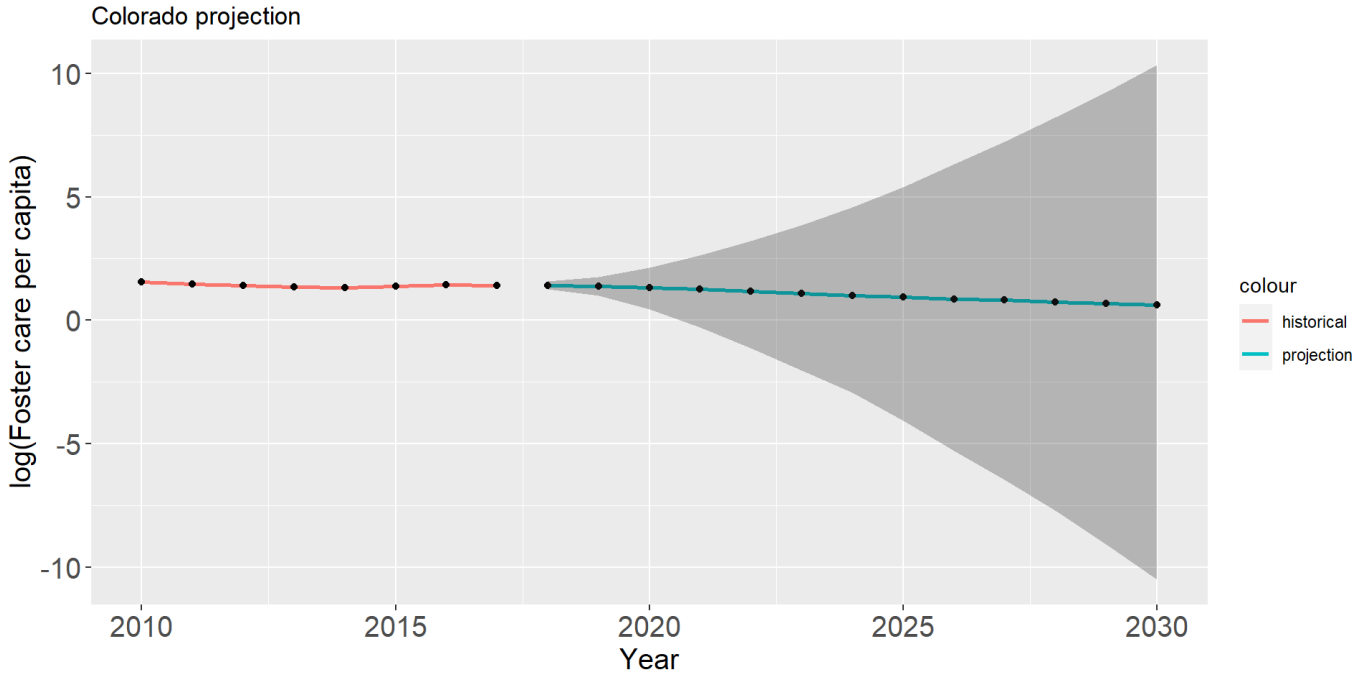
Projections for the state of Florida

```
state_name = "Florida"
state_id = which(states == state_name)
florida <- calc_dstats(y_proj[, , state_id])
florida_historic <- d %>%
    filter(state == state_name)

ggplot(data = florida) + geom_line(aes(x = years, y = median, color = "projection"),
    linewidth = 1) + geom_point(aes(x = years, y = median)) + geom_ribbon(mapping = aes(x = y
ears,
    ymin = lb, ymax = ub), alpha = 0.3) + ggtitle("Florida projection") + geom_line(data = fl
orida_historic,
    aes(x = years, y = log(ent_pc), color = "historical"), linewidth = 1) + geom_point(data =
florida_historic,
    aes(x = years, y = log(ent_pc))) + theme_large_text + labs(x = "Year", y = "log(Foster ca
re per capita)")
```



Projections for the state of Colorado

```
state_name = "Colorado"
state_id = which(states == state_name)
colorado <- calc_dstats(y_proj[, , state_id])
colorado_historic <- d %>%
    filter(state == state_name)

ggplot(data = colorado) + geom_line(aes(x = years, y = median, color = "projection"),
    linewidth = 1) + geom_point(aes(x = years, y = median)) + geom_ribbon(mapping = aes(x = y
ears,
    ymin = lb, ymax = ub), alpha = 0.3) + ggtitle("Colorado projection") + geom_line(data = c
olorado_historic,
    aes(x = years, y = log(ent_pc), color = "historical"), linewidth = 1) + geom_point(data =
colorado_historic,
    aes(x = years, y = log(ent_pc))) + theme_large_text + labs(x = "Year", y = "log(Foster ca
re per capita)")
```



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?

Answer: Introducing more covariates may cause multicollinearity or overfitting. Overfitting may be alleviated using regularization similar to the term in ridge regression. Multicollinearity may be solved by doing variable selection to remove highly correlated variables.