Bayesian Rogers-Castro models for migration

BMSS, October 16 2024

Monica Alexander

Table of contents

1 Overview

This Quarto document illustrates how to fit Rogers-Castro models to age-specific migration rates using the `rcbayes` R package. Here's a paper that gives some more information the package.

Load in the required packages:

```
library(tidyverse)
library(rcbayes)
```

2 What is a Rogers-Castro model?

The Rogers-Castro model (1981) for migration age schedules is multi-exponential parametric model that aims to capture the non-linear characteristic non-linear shape of migration. It has the form:

$$
\begin{aligned}
m(x) = & a_1 \exp\{-\alpha_1 x\} + \\
& a_2 \exp\{-\alpha_2 (x - \mu_2) - \exp[-\lambda_2 (x - \mu_2)]\} + \\
& a_3 \exp\{-\alpha_3 (x - \mu_3) - \exp[-\lambda_3 (x - \mu_3)]\} + \\
& a_4 \exp\{\alpha_4 x\} + \\
& c
\end{aligned}
$$

where $m(x)$ is the migration rate (in- or out-) for age x. The successive additive components of the model represent pre-working, working, retirement, post-retirement, and overall migration, respectively. Various versions of the model can be estimated; for example, a simpler version would remove the components related to retirement and post-retirement peaks. The `rcbayes` package has a built-in interaction Shiny application that you can use to explore how different parameter values affect the shape of the migration age curve:

```r
rcbayes::interact_rc()
```

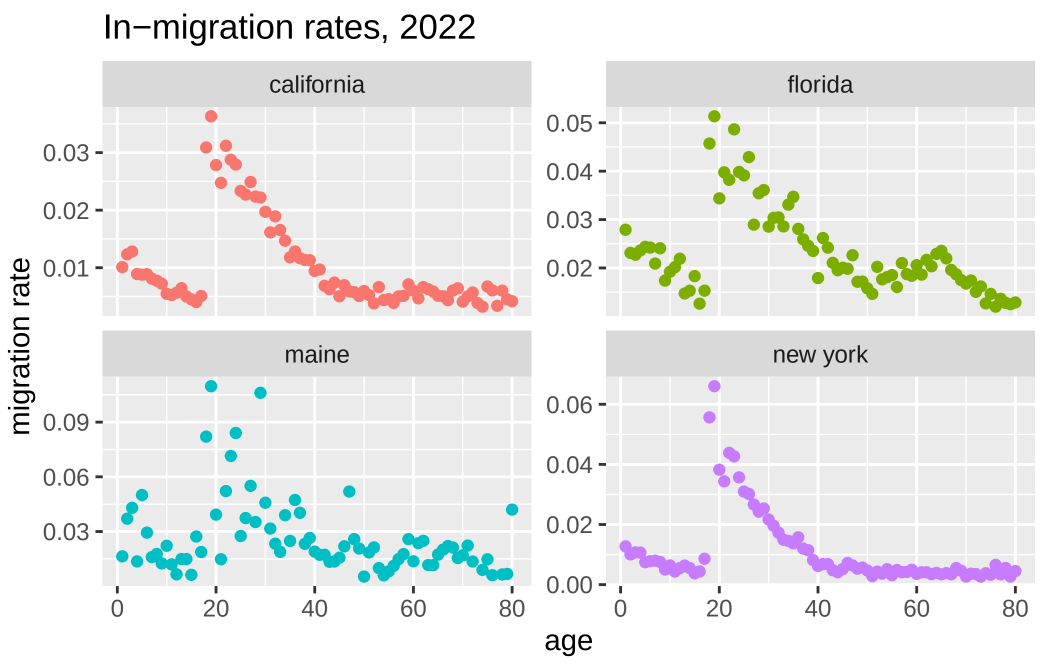
3 Read in data

For this example, we'll be using data from the 2022 round of the American Community Survey (ACS). The data below show the population by age and number of in-migrants in the past year by age for each US state.

```r
d <- read_csv("../data/us_state_mig.csv")
```

We can plot the in-migration rates observed in the ACS for a few different states. Florida has a clear retirement peak, which is not obvious in the other examples. Maine, which is a smaller state, has much noisier data.

```r
d |>
  filter(state %in% c("florida", "new york", "california", "maine")) |>
  ggplot(aes(age, rate, color = state)) +
  geom_point()+
  facet_wrap(~state, scales = "free_y")+
  theme(legend.position = "none")+
  labs(title = "In-migration rates, 2022", y = "migration rate")
```



4 Fit Rogers-Castro model

Now we're going to fit a Rogers-Castro migration curve to some ACS data, starting with Florida. To do this, we're going to use the `mig_estimate_rc` function in the `rcbayes` package. This is a wrapper function for `rstan`, and allows you to specify which components of the curve to fit, as well as the usual Stan control settings. For Florida, let's fit the full 13 parameter model.

```r
d_florida <- d |> filter(state=="florida")
ages <- d_florida$age
migrants <- d_florida$n_mig
pop <-  d_florida$pop

res <- rcbayes::mig_estimate_rc(ages = ages,
                                migrants = migrants,
                                pop = pop,
                                pre_working_age = TRUE,
                                working_age = TRUE,
                                retirement = TRUE,
                                post_retirement = TRUE,
                                control = list(adapt_delta = 0.96),
                                seed = 1899)
```

```
SAMPLING FOR MODEL 'rcmodel_poisson' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 9.9e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.99 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: 6.82618 seconds (Warm-up)
Chain 1:                11.4757 seconds (Sampling)
Chain 1:                18.3018 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'rcmodel_poisson' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.4e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.44 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: 8.40788 seconds (Warm-up)
Chain 2:                6.946 seconds (Sampling)
Chain 2:                15.3539 seconds (Total)
Chain 2: 

SAMPLING FOR MODEL 'rcmodel_poisson' NOW (CHAIN 3).
Chain 3: 
Chain 3: Gradient evaluation took 7.4e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.74 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: 9.09338 seconds (Warm-up)
Chain 3:                7.11401 seconds (Sampling)
Chain 3:                16.2074 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL 'rcmodel_poisson' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 4.5e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.45 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: 7.79391 seconds (Warm-up)
Chain 4:                6.87411 seconds (Sampling)
Chain 4:                14.668 seconds (Total)
Chain 4:
```

The `res` object has several elements, including a summary of the model parameters, convergence checks, and the estimated fit and uncertainty.

```
names(res)
```

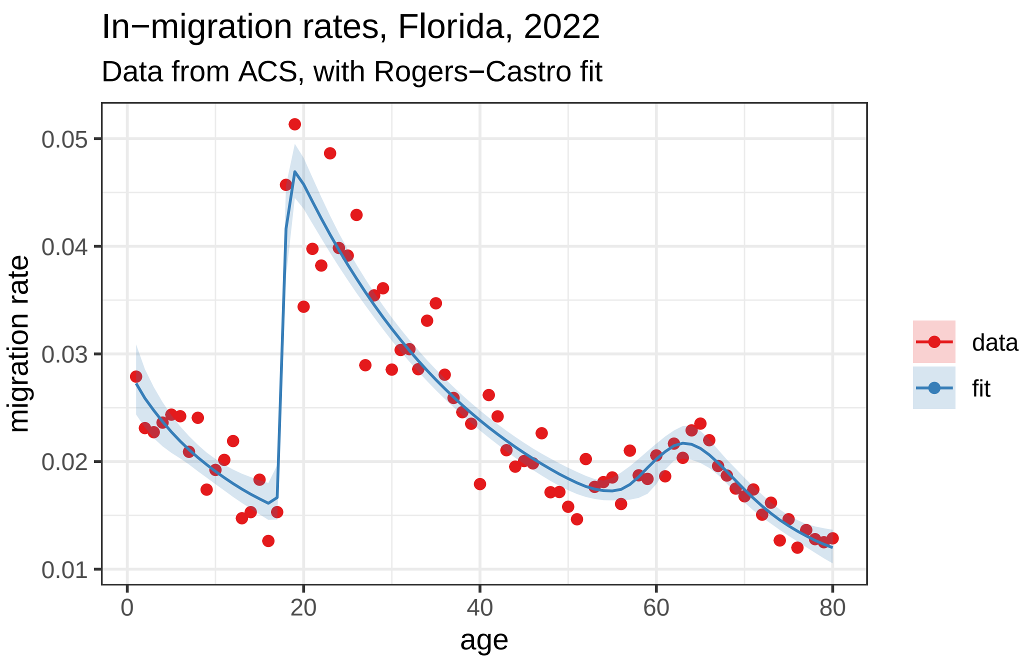
```
[1] "pars_df"        "fit_df"         "check_converge"
```

```
res$pars_df
```

```
# A tibble: 13 x 4
   variable   median     lower    upper
   <chr>       <dbl>     <dbl>    <dbl>
 1 a1        0.0171   0.00533   0.0270
 2 a2        0.0347   0.0311    0.0382
 3 a3        0.0208   0.0150    0.0268
 4 a4        0.00693  0.000317  0.0222
 5 alpha1    0.0797   0.0407    1.57
 6 alpha2    0.0443   0.0322    0.0587
 7 alpha3    0.202    0.118     0.387
 8 c         0.00432  0.000196  0.0104
 9 lambda2   2.80     1.99      3.94
10 lambda3   0.172    0.119     0.295
11 lambda4  -0.00780 -0.0291    0.00937
12 mu2      17.5     17.3      17.7
13 mu3      64.9     63.0      66.8
```

Let's pull out the fit dataframe (which also have the data), and plot to see what it looks like.

```
res$fit_df |>
  ggplot(aes(age, data)) +
  geom_point(aes(color = "data", fill = "data"))+
  geom_line(aes(age, median, color = "fit"))+
  geom_ribbon(aes(age, ymin = lower, ymax = upper, fill = "fit"), alpha = 0.2)+
  scale_color_brewer(name = "", palette = "Set1")+
  scale_fill_brewer(name = "", palette = "Set1")+
  theme_bw()+
  labs(title = "In-migration rates, Florida, 2022", subtitle = "Data from ACS, with Rogers
```



4.1 Another example XX to do

5 Extensions

The `rcbayes` package was designed for users to be able to fit Rogers-Castro models to a single population. The Stan code for the models in the packages is here. These could be extended to fit, for example, hierarchical models for multiple areas at a time.