Introduction to Bayesian Models for Demographers

Summer Incubator Workshop, 25 June 2024

Monica Alexander

Table of contents

1 Overview

This Quarto document illustrates how to fit a Gompertz mortality model in a Bayesian framework using Stan, with a couple of extensions. We will be using data from the Canadian HMD, and some simulated data, as an example.

2 Packages required

To follow along and execute the code on your own computer, you will need the packages below installed and loaded. If you don't have them installed, you can do so by using `install.packages()`. The exception is `rstan`, which can be a bit tricky to get working; detailed instructions on how to install can be found here.

```
library(rstan)
library(rstanarm)
library(tidyverse)
library(tidybayes)
library(janitor)
```

3 Data

For the following two examples we're going to using death and population counts by age and sex for Ontario, sourced from the Canadian Human Mortality Database project. We can read the data files in directly from the URLs:

```
dd <- read_table("https://www.prdh.umontreal.ca/BDLC/data/ont/Deaths_1x1.txt", skip = 1)
dp <- read_table("https://www.prdh.umontreal.ca/BDLC/data/ont/Population.txt", skip = 1)
```

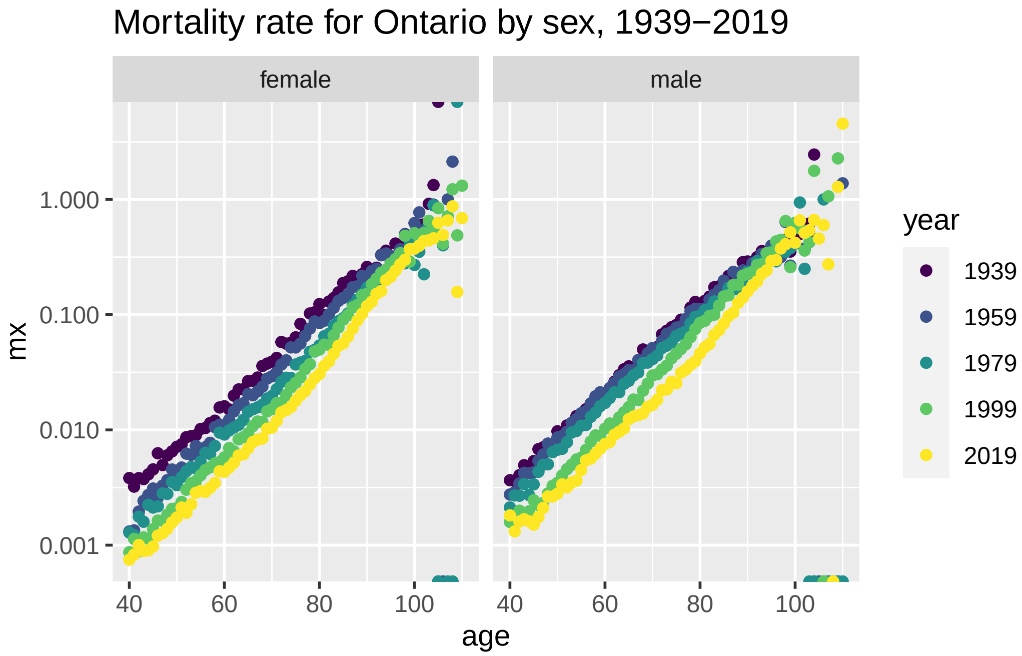
These data files are in 'wide' format. For our purposes it's going to be easier to work with in 'long format'. So let's do that and also clean some other stuff up:

```
dd <- dd |>
  clean_names() |>
  pivot_longer(-(year:age), names_to = "sex", values_to = "deaths") |>
  mutate(deaths = as.numeric(deaths), age = as.numeric(age)) |>
  mutate(age = ifelse(is.na(age), 110, age))

dp <- dp |>
```

```
  clean_names() |>
  pivot_longer(-(year:age), names_to = "sex", values_to = "pop") |>
  mutate(pop = as.numeric(pop), age = as.numeric(age)) |>
  mutate(age = ifelse(is.na(age), 110, age))

d <- dd |>
  left_join(dp) |>
  mutate(mx = deaths/pop,
         log_mx = log(mx))
```

Do some quick plots

```
d |>
  filter(age>39, year %in% seq(1939, 2019, by = 20), sex!= "total") |>
  ggplot(aes(age, mx, color = factor(year))) +
  geom_point() +
  facet_wrap(~sex)+
  labs(title = "Mortality rate for Ontario by sex, 1939-2019")+
  scale_y_log10()+
  scale_color_viridis_d(name = "year")
```



4 Base model

First let's fit a Gompertz model to males aged 40+ in 2019. Note that Gompertz models have the form

$$m_x = \alpha e^{\beta x}$$

So we can write

$$\log m_x = \log \alpha + \beta x$$

More notes:

- To make this fully Bayesian we need to specify the likelihood and priors. The full model here is

$$y_x \sim \text{Poisson}(P_x \cdot m_x)$$

$$\log m_x = \log \alpha + \beta x$$

$$\log \alpha \sim N(0, 10^2)$$

$$\beta \sim N(0, 0.1^2)$$

  where y_x is deaths at age x and P_x is population
- Could have used a normal likelihood (c.f. using `lm`) but nice to account for population size
- We are fitting not on age, but on a centered version (why?)

Now we need to get the data in the right format to read into Stan (this required a named list):

```r
d_male_19 <- d |>
  filter(year==2019, sex == "male", age>39, age<105) |>
  mutate(age_c = age - 40)

stan_data <- list(y = round(d_male_19$deaths),
                  pop = round(d_male_19$pop),
                  N = nrow(d_male_19),
                  age_c = d_male_19$age_c)
```

Run the model and look at some output:

```r
mod <- stan(file = "models/gomp.stan",
            data = stan_data,
            seed = 123,
            refresh = 0)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDE
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEig
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEig
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/C
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/C
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEig
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

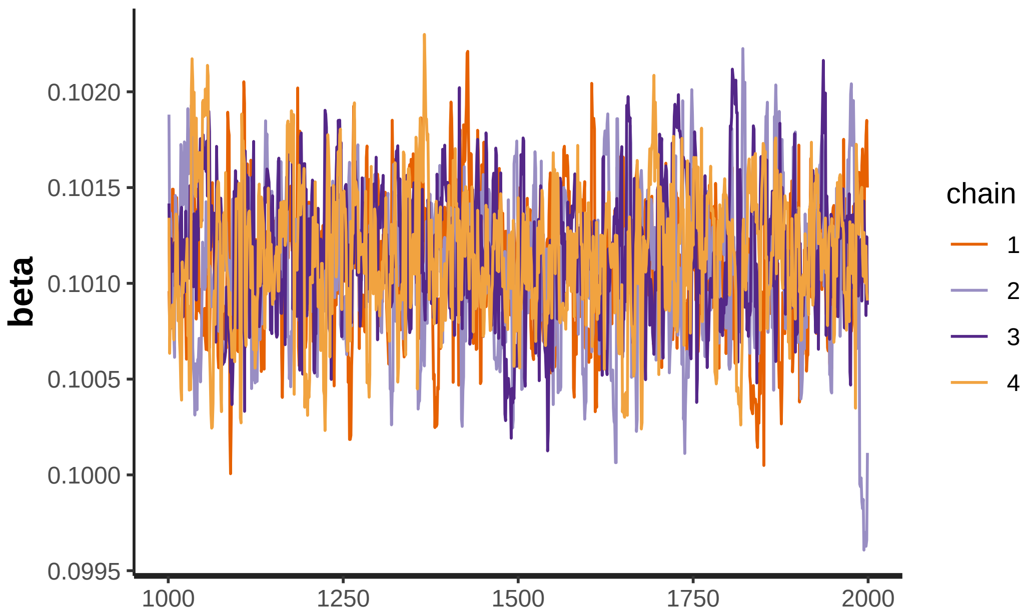
```r
summary(mod)$summary[c("log_alpha", "beta"),]
```

```
                mean      se_mean           sd       2.5%         25%         50%
log_alpha -7.0080473 5.314773e-04 0.0139930761 -7.0360337 -7.0174794 -7.0082219
beta       0.1011261 1.365872e-05 0.0003643962  0.1004042  0.1008886  0.1011334
                75%      97.5%    n_eff     Rhat
log_alpha -6.998623 -6.9804263 693.1973 1.007995
beta       0.101374  0.1018417 711.7498 1.007239
```

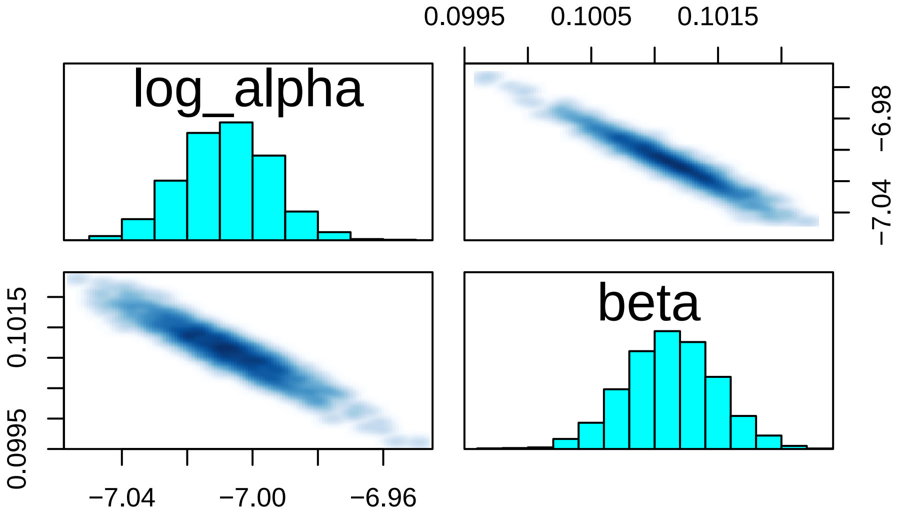
Some quick model checks:

- A traceplot plots the sampled values for a particular parameter across iterations. For a well mixing MCMC chain, you want it to look like a 'fuzzy caterpillar'
- The pairs plot shows you the joint distribution of the sampled parameter values, and can be useful to diagnose problems in particular parts of the parameter space

```r
pars <- c("log_alpha", "beta")
traceplot(mod, pars = c("beta"))
```



```r
pairs(mod, pars = pars)
```



4.1 Help! I'm drowning in samples

What is the output we're actually interested in? The posterior samples. The 'classic' way of extracting using functions from the rstan package:

```
samps <- rstan::extract(mod)
length(samps)
```

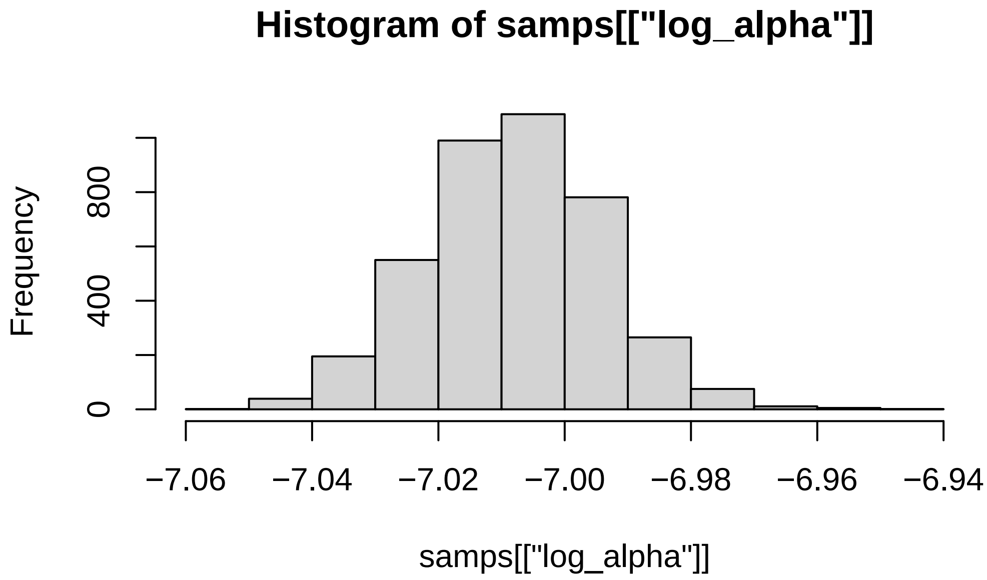
```
[1] 4
```

```
names(samps)
```

```
[1] "log_alpha" "beta"      "mu"        "lp__"
```

Inference about the parameters of interest is now done on the samples:

```
hist(samps[["log_alpha"]])
```



```
mean(samps[["log_alpha"]]); quantile(samps[["log_alpha"]], 0.975); quantile(samps[["log_al
[1] -7.008047
```

```
    97.5%
-6.980426
```

```
     2.5%
-7.036034
```

4.2 tidybayes makes it easier

Cleaner in the tiydverse style grammar:

```
mod |>
  gather_draws(log_alpha, beta) |>
  median_qi()
```

```
# A tibble: 2 x 7
  .variable .value .lower .upper .width .point .interval
  <chr>      <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
1 beta       0.101  0.100  0.102   0.95 median qi
2 log_alpha -7.01  -7.04  -6.98    0.95 median qi
```

4.3 Side note: rstanarm is good for these simpler models

Above, we wrote our own Stan model to fit a Gompertz model to one year. This is probably a bit of overkill (although good to see). The rstanarm and brms packages are very useful for standard models (if you've used lme4, the syntax is similar, including for multilevel models). For example here's the same model fit in rstanarm:

```
mod_pois_rsarm <- stan_glm(deaths ~ age_c + offset(log(pop)),
                     data = d_male_19,
                     family = poisson,
                     refresh = 0)

summary(mod_pois_rsarm)
```

```
Model Info:
 function:     stan_glm
 family:       poisson [log]
 formula:      deaths ~ age_c + offset(log(pop))
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 65
 predictors:   2

Estimates:
              mean   sd   10%   50%   90%
(Intercept) -7.0    0.0 -7.0  -7.0  -7.0
age_c        0.1    0.0  0.1   0.1   0.1

Fit Diagnostics:
           mean   sd    10%   50%   90%
mean_PPD 808.5    5.0 802.1 808.7 814.7

The mean_ppd is the sample average posterior predictive distribution of the outcome variable

MCMC diagnostics
              mcse Rhat n_eff
(Intercept)   0.0  1.0  2221
age_c         0.0  1.0  2384
mean_PPD      0.1  1.0  2862
log-posterior 0.0  1.0  1638

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effectiv
```

4.4 Side note 2: sanity check

Check our Bayes results against standard GLM:

```
summary(lm(log_mx~age_c, data = d_male_19))
```

```

Call:
lm(formula = log_mx ~ age_c, data = d_male_19)

Residuals:
     Min       1Q   Median       3Q      Max
-0.27297 -0.11044 -0.00203  0.07829  0.58094

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.894031   0.039061  -176.5   <2e-16 ***
age_c        0.099829   0.001053    94.8   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1593 on 63 degrees of freedom
Multiple R-squared:  0.993, Adjusted R-squared:  0.9929
F-statistic:  8988 on 1 and 63 DF,  p-value: < 2.2e-16
```

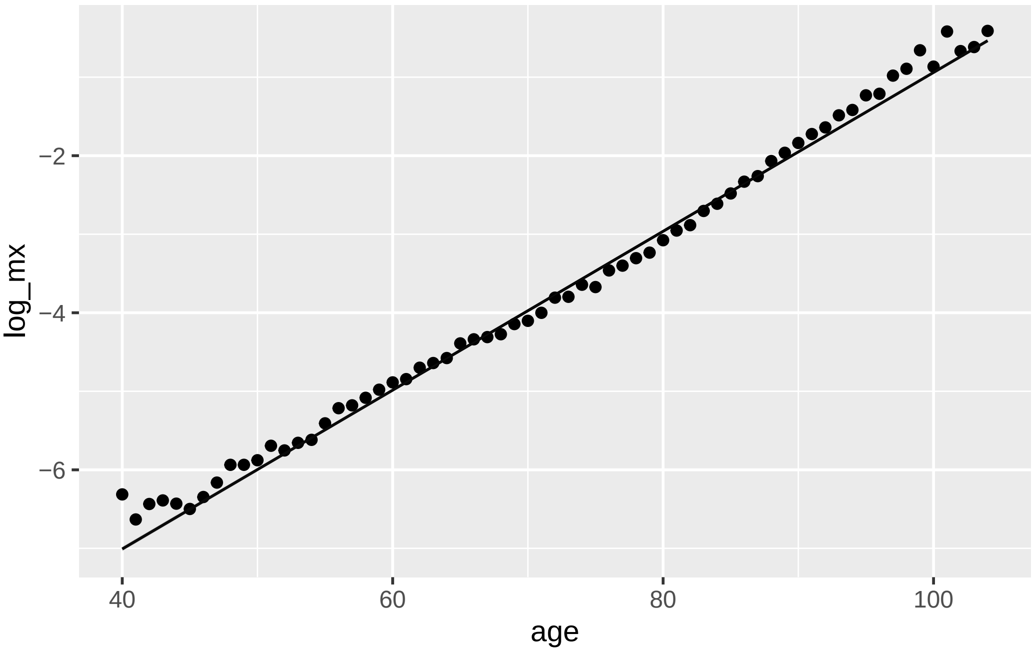
```
summary(glm(deaths~age_c+offset(log(pop)), family = poisson, data = d_male_19))
```

```
Call:
glm(formula = deaths ~ age_c + offset(log(pop)), family = poisson,
    data = d_male_19)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-7.292  -1.335   1.206   3.507   7.854

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.0078675  0.0136705  -512.6   <2e-16 ***
age_c        0.1011213  0.0003567   283.5   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 91447.1  on 64  degrees of freedom
Residual deviance:   767.1  on 63  degrees of freedom
AIC: 1299.7

Number of Fisher Scoring iterations: 4
```

4.5 Calculate and plot some results

Now we can combine the `tidybayes` syntax with `ggplot` to plot some results. For example, here's the data versus the estimates for the linear predictor:

```
mod |>
  gather_draws(mu[x]) |>
  median_qi() |>
  mutate(age_c = x - 1) |>
  left_join(d_male_19) |>
  ggplot(aes(age, log_mx)) +
  geom_point() +
  geom_line(aes(age, .value)) +
  geom_ribbon(aes(x = age, ymin = .lower, ymax = .upper), alpha = 0.2)
```



A neat thing about working with samples is that we can easily get estimates and uncertainty for a transformed version of our parameters. For example, let's calculate the modal age at death (which for Gompertz mortality is a function of α and β, see here).

```
mod |>
  spread_draws(log_alpha, beta) |>
  mutate(mode_age = 1/beta*log(beta/exp(log_alpha))) |>
  median_qi()
```

```
# A tibble: 1 x 12
  log_alpha log_alpha.lower log_alpha.upper  beta beta.lower beta.upper mode_age
```

```
      <dbl>           <dbl>           <dbl> <dbl>      <dbl>      <dbl>    <dbl>
1     -7.01           -7.04           -6.98 0.101      0.100      0.102      46.6
# i 5 more variables: mode_age.lower <dbl>, mode_age.upper <dbl>, .width <dbl>,
#   .point <chr>, .interval <chr>
```

4.6 Question for you

Open the Stan model and edit the priors to be more informative. What do you expect to happen to the resulting estimates? What happens if you set very informative, but nonsensical priors? (e.g., $\beta \sim N(0.5, 0.001^2)$)

5 Model over time

Let's fit a slightly more complicated model, for multiple years, where the coefficients themselves are modeled as a random walk over time, i.e.

$$\beta_t \sim N(\beta_{t-1}, \sigma_\beta^2)$$

That is, a different set of Gompertz parameters are fit to every year, but we are assuming that the values in the current year are related to those is the previous year. This is a form of dynamic linear regression. We need to put priors on the first time point, and also on the variance terms:

$$\log \alpha_1 \sim N(-6, 1)$$
$$\beta_1 \sim N(0.1, 0.1^2)$$
$$\sigma^{\alpha}, \sigma_\beta \sim N^+(0, 1)$$

Note that the likelihood and model on mortality rates are as before, we just have an additional subscript for time:

$$y_{x,t} \sim \text{Poisson}(P_{x,t} \cdot m_{x,t})$$
$$\log m_{x,t} = \log \alpha_t + \beta_t x$$

Now to fit the model. First get the data in the right format:

```r
years <- 1969:2019
d_male <- d |> filter(year>=years[1], sex == "male", age>39, age<105) |>
  mutate(age_c = age - 40)

y <- d_male |>
  select(age, year, deaths) |>
  pivot_wider(names_from = "year", values_from = "deaths") |>
  select(-age) |>
  as.matrix()

pop <- d_male |>
  select(age, year, pop) |>
  pivot_wider(names_from = "year", values_from = "pop") |>
  select(-age) |>
  as.matrix()

stan_data <- list(y = y,
                  pop = pop,
                  N = nrow(d_male_19),
                  age_c = d_male_19$age_c,
                  T = ncol(y))
```

Now fit the model (note: takes a while):

```
mod <- stan(file = "models/gomp_time.stan",
            data = stan_data,
            seed = 852,
            refresh = 0)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEI
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:9
#include <complex>
         ^~~~~~~~~
3 errors generated.
```

```
make: *** [foo.o] Error 1
```

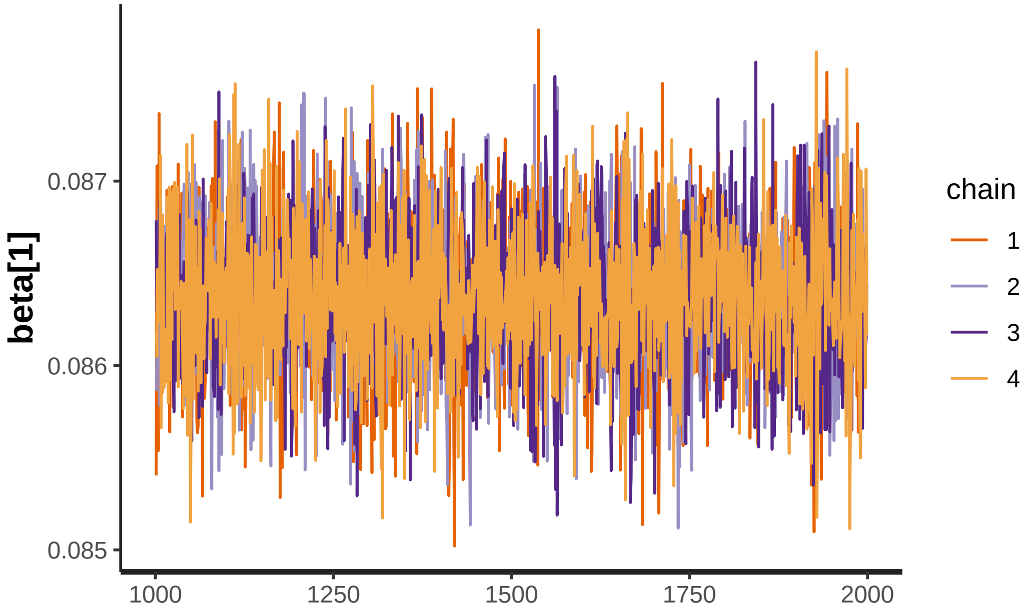
```
summary(mod)$summary[paste0("beta[", 1:(length(years)), "]"),]
```

```
               mean      se_mean           sd       2.5%        25%        50%
beta[1]  0.08637726 5.408155e-06 0.0003980909 0.08558443 0.08611616 0.08637743
beta[2]  0.08609944 5.382601e-06 0.0003651452 0.08538432 0.08585654 0.08609603
beta[3]  0.08652996 5.348717e-06 0.0003544698 0.08583556 0.08629793 0.08653449
beta[4]  0.08684064 5.400215e-06 0.0003564779 0.08612462 0.08659735 0.08684537
beta[5]  0.08741744 5.199602e-06 0.0003509300 0.08673113 0.08718470 0.08740712
beta[6]  0.08748773 4.949059e-06 0.0003538722 0.08677683 0.08724562 0.08748556
beta[7]  0.08774906 5.227438e-06 0.0003470942 0.08707205 0.08752175 0.08774641
beta[8]  0.08775315 4.605522e-06 0.0003488372 0.08706722 0.08751955 0.08775056
beta[9]  0.08714442 4.909790e-06 0.0003637462 0.08642407 0.08690333 0.08715067
beta[10] 0.08750291 5.067639e-06 0.0003493777 0.08683635 0.08726701 0.08749992
beta[11] 0.08849544 5.278865e-06 0.0003562111 0.08778669 0.08825092 0.08850578
beta[12] 0.08985115 4.954444e-06 0.0003511657 0.08917759 0.08960734 0.08985032
beta[13] 0.09052507 4.938765e-06 0.0003514422 0.08981549 0.09029087 0.09052456
beta[14] 0.09140559 5.018458e-06 0.0003498535 0.09074796 0.09116877 0.09140231
beta[15] 0.09215278 5.256351e-06 0.0003468065 0.09148419 0.09192024 0.09214732
beta[16] 0.09236948 5.350142e-06 0.0003474065 0.09170140 0.09213687 0.09236870
beta[17] 0.09318506 4.934216e-06 0.0003427723 0.09250845 0.09296097 0.09318540
beta[18] 0.09362172 4.827602e-06 0.0003407646 0.09294258 0.09338867 0.09362186
beta[19] 0.09448456 5.031244e-06 0.0003450523 0.09380537 0.09425529 0.09449205
beta[20] 0.09589743 4.736623e-06 0.0003380852 0.09523916 0.09567167 0.09589801
beta[21] 0.09570937 4.632706e-06 0.0003402578 0.09504408 0.09547927 0.09570833
beta[22] 0.09559315 5.000297e-06 0.0003473280 0.09490788 0.09536296 0.09558930
beta[23] 0.09665824 4.950162e-06 0.0003413273 0.09601284 0.09641574 0.09665355
beta[24] 0.09687286 5.376057e-06 0.0003385324 0.09621242 0.09664052 0.09687314
beta[25] 0.09707877 4.906067e-06 0.0003410519 0.09639510 0.09685583 0.09708573
beta[26] 0.09803144 4.597040e-06 0.0003342023 0.09736212 0.09780229 0.09803443
beta[27] 0.09873372 4.623289e-06 0.0003342355 0.09808297 0.09849884 0.09873116
beta[28] 0.09982051 4.129549e-06 0.0003178325 0.09921858 0.09960067 0.09981570
beta[29] 0.10069637 4.539651e-06 0.0003259581 0.10004710 0.10047692 0.10069030
beta[30] 0.10201285 4.426542e-06 0.0003303988 0.10138374 0.10179314 0.10200705
beta[31] 0.10278899 4.890553e-06 0.0003238709 0.10216854 0.10257216 0.10278989
beta[32] 0.10218301 4.836046e-06 0.0003368970 0.10151708 0.10195512 0.10218451
beta[33] 0.10154374 4.535083e-06 0.0003290753 0.10088974 0.10132732 0.10154684
beta[34] 0.10109115 4.577365e-06 0.0003332883 0.10044712 0.10086770 0.10109366
beta[35] 0.10134945 4.449425e-06 0.0003206309 0.10071706 0.10113792 0.10134655
beta[36] 0.10127201 4.370720e-06 0.0003215180 0.10062505 0.10105768 0.10127816
```

```
beta[37] 0.10170418 4.901921e-06 0.0003188131 0.10109436 0.10148031 0.10170576
beta[38] 0.10161387 4.433367e-06 0.0003270773 0.10096578 0.10139755 0.10161475
beta[39] 0.10157482 4.326682e-06 0.0003197609 0.10092946 0.10136405 0.10157236
beta[40] 0.10205767 4.174372e-06 0.0003099799 0.10145796 0.10184443 0.10206079
beta[41] 0.10201268 4.166429e-06 0.0003111773 0.10139494 0.10180401 0.10200843
beta[42] 0.10192130 4.156107e-06 0.0003146041 0.10131034 0.10170127 0.10192421
beta[43] 0.10195703 4.263020e-06 0.0003134395 0.10136025 0.10174919 0.10195307
beta[44] 0.10183627 4.423804e-06 0.0003147932 0.10123076 0.10161419 0.10182999
beta[45] 0.10213364 4.067378e-06 0.0003045724 0.10153586 0.10192837 0.10213084
beta[46] 0.10230145 4.138734e-06 0.0002977424 0.10170537 0.10210635 0.10229703
beta[47] 0.10218170 4.094610e-06 0.0003122484 0.10158453 0.10197153 0.10217805
beta[48] 0.10136463 4.190264e-06 0.0003045539 0.10076529 0.10115875 0.10137030
beta[49] 0.10180627 4.051448e-06 0.0002999508 0.10121524 0.10160704 0.10180346
beta[50] 0.10137754 4.156122e-06 0.0002967507 0.10078747 0.10116896 0.10138211
beta[51] 0.10076741 4.196141e-06 0.0002926109 0.10020392 0.10056956 0.10076230
                75%      97.5%    n_eff      Rhat
beta[1]  0.08663952 0.08715981 5418.341 0.9997651
beta[2]  0.08634806 0.08681256 4602.004 0.9996388
beta[3]  0.08676876 0.08721581 4391.971 0.9993949
beta[4]  0.08708340 0.08750984 4357.558 0.9994744
beta[5]  0.08765920 0.08811718 4555.130 0.9994284
beta[6]  0.08772717 0.08817612 5112.668 0.9998507
beta[7]  0.08799186 0.08841994 4408.764 1.0005681
beta[8]  0.08798697 0.08844275 5737.038 0.9997367
beta[9]  0.08739520 0.08783403 5488.722 0.9995110
beta[10] 0.08773899 0.08819625 4753.123 0.9992677
beta[11] 0.08874090 0.08917115 4553.379 0.9997617
beta[12] 0.09009148 0.09054310 5023.823 0.9998000
beta[13] 0.09075470 0.09121383 5063.736 0.9994911
beta[14] 0.09164203 0.09211567 4859.950 1.0000228
beta[15] 0.09239278 0.09281552 4353.171 1.0000687
beta[16] 0.09260002 0.09306442 4216.434 1.0008383
beta[17] 0.09341342 0.09384493 4825.865 1.0005281
beta[18] 0.09385106 0.09430672 4982.486 0.9995901
beta[19] 0.09471497 0.09516511 4703.477 0.9994943
beta[20] 0.09612373 0.09655150 5094.655 0.9994698
beta[21] 0.09594115 0.09638838 5394.442 1.0008745
beta[22] 0.09582528 0.09627444 4824.898 0.9998200
beta[23] 0.09689225 0.09734074 4754.480 0.9996771
beta[24] 0.09710082 0.09754250 3965.270 0.9997944
beta[25] 0.09730369 0.09774547 4832.524 0.9990396
beta[26] 0.09824938 0.09870363 5285.211 0.9997612
beta[27] 0.09897322 0.09939630 5226.407 0.9992504
```

```
beta[28] 0.10003862 0.10044991 5923.678 0.9998480
beta[29] 0.10091535 0.10134433 5155.595 0.9998185
beta[30] 0.10223717 0.10266737 5571.187 0.9995604
beta[31] 0.10299817 0.10343151 4385.591 0.9995600
beta[32] 0.10240877 0.10284245 4853.035 0.9993742
beta[33] 0.10176144 0.10218559 5265.265 0.9993428
beta[34] 0.10130965 0.10174405 5301.625 0.9995127
beta[35] 0.10156371 0.10196972 5192.815 0.9994740
beta[36] 0.10149341 0.10189053 5411.338 1.0003346
beta[37] 0.10192093 0.10233315 4229.993 1.0003420
beta[38] 0.10184554 0.10223206 5442.936 1.0001457
beta[39] 0.10179104 0.10218890 5461.861 0.9993711
beta[40] 0.10226643 0.10267457 5514.229 1.0000402
beta[41] 0.10222812 0.10261621 5578.119 0.9997010
beta[42] 0.10213817 0.10253942 5730.006 0.9996566
beta[43] 0.10216582 0.10258639 5405.959 0.9996635
beta[44] 0.10205022 0.10244746 5063.596 1.0001324
beta[45] 0.10233665 0.10272931 5607.279 0.9998480
beta[46] 0.10250437 0.10287845 5175.427 0.9991479
beta[47] 0.10238781 0.10281103 5815.343 0.9996321
beta[48] 0.10157612 0.10194778 5282.576 1.0005244
beta[49] 0.10200775 0.10240731 5481.251 1.0001391
beta[50] 0.10158540 0.10195414 5098.084 1.0003962
beta[51] 0.10096640 0.10134141 4862.738 0.9994851
```

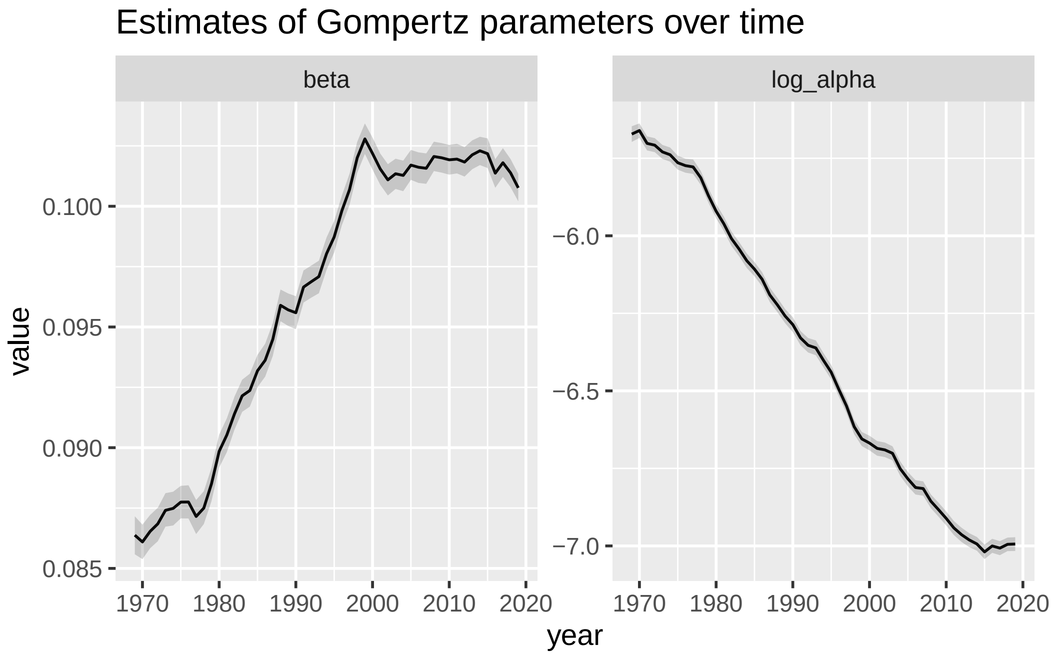
```r
traceplot(mod, pars = c("beta[1]"))
```



5.1 Plot parameter estimates over time

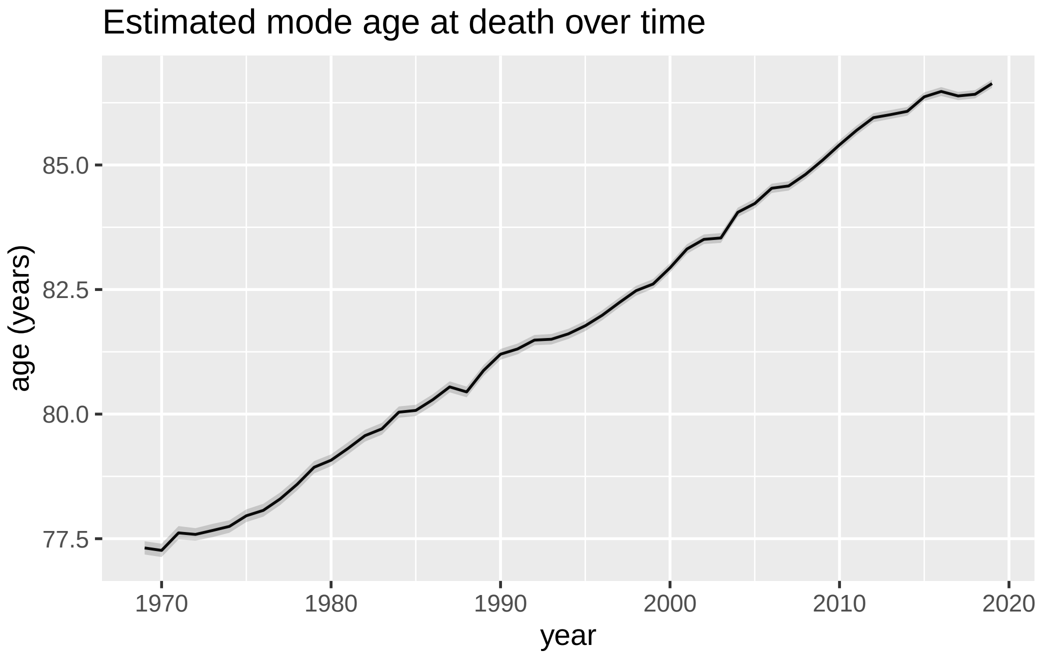
Now we can plot the parameter estimates over time:

```r
mod |>
  gather_draws(log_alpha[i], beta[i]) |>
  median_qi() |>
  mutate(year = years[i]) |>
  ggplot(aes(year, .value)) + geom_line() +
  facet_wrap(~.variable, scales = "free_y") +
  geom_ribbon(aes(ymin = .lower, ymax = .upper), alpha = 0.2)+
  labs(y = "value", title = "Estimates of Gompertz parameters over time")
```



…and also the mode age over time:

```
mod |>
  spread_draws(log_alpha[i], beta[i]) |>
  mutate(mode_age = 1/beta*log(beta/exp(log_alpha))+40) |>
  median_qi() |>
  mutate(year = years[i]) |>
  ggplot(aes(year, mode_age)) + geom_line() +
  geom_ribbon(aes(ymin = mode_age.lower, ymax = mode_age.upper), alpha = 0.2) +
  labs(title = "Estimated mode age at death over time", y = "age (years)")
```



5.2 Question for you

Can you forecast mortality rates with this model? If so, how? What are the assumptions behind the forecasts?

6 Partial data observed

Now we're going to switch gears a bit and have a look at the situation where we have mortality rates for five geographic areas, but in one area we only have partial information. This is based on simulated data (you can have a look to see how I generated it based on the `simulated_data.R` script).

For 4 areas, we have deaths and population counts from ages 40 up to 60:

```
df <- read_rds("data/sim.rds")
df
```

```
# A tibble: 80 x 4
# Groups:   area [4]
     age  area deaths   pop
   <int> <int>  <int> <dbl>
```

```
 1    40     1      1  5000
 2    41     1      1  5000
 3    42     1      1  5000
 4    43     1      2  5000
 5    44     1      1  5000
 6    45     1      2  5000
 7    46     1      0  5000
 8    47     1      3  5000
 9    48     1      5  5000
10    49     1      3  5000
# i 70 more rows
```

For one region, we just have ${}_{20}q_{40}$, that is, the probability of dying between ages 40 and 60:

```
q40 <- read_rds("data/q40.rds")
q40
```

```
[1] 0.07262754
```

6.1 Relating ${}_{20}q_{40}$ to the Gompertz model

We want to fit a Gompertz model to each of the five areas (even the one with just a summary indicator). How to do this? Well, if we tell Stan how ${}_{20}q_{40}$ relates to μ_x, then the model has at least some information to estimate α and β. In particular, for each area, we're assuming:

$$ {}_{1}p_x = e^{-\mu_x} $$

where $\mu_x = \alpha e^{\beta x}$ and

$$ {}_{20}q_{40} = 1 - \prod_{x=40}^{60} {}_{1}p_x $$

Check the Stan file `gomp_partial.stan` to see this translated into code. ## Fit the model

Get the data in the right format. Note that everything is a matrix now because we have more than one area:

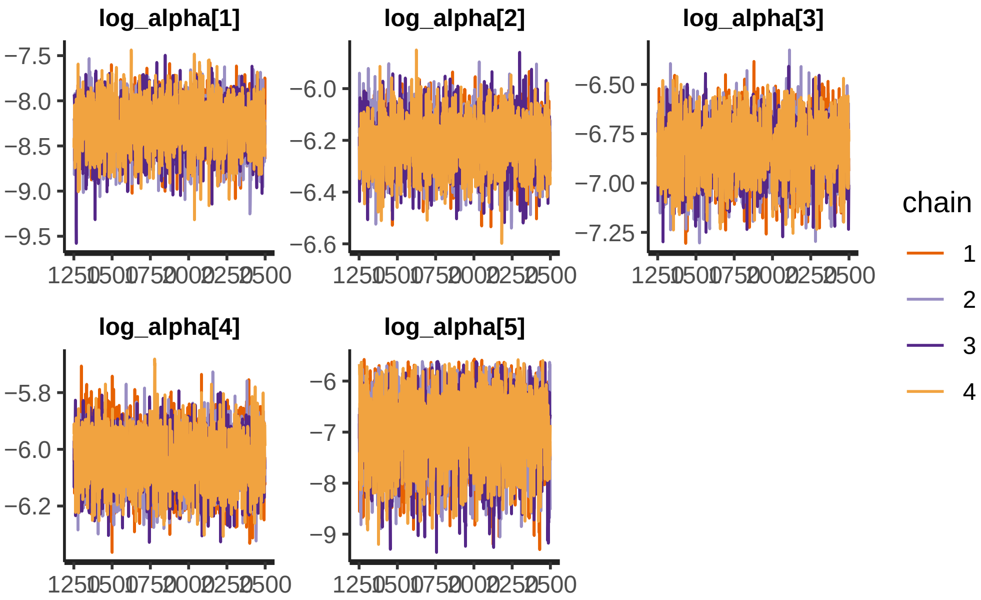
```
y <- as.matrix(df |>
  pivot_wider(names_from = "area", values_from = "deaths") |>
  select(-age, -pop))
pop <- as.matrix(df |>
  select(-deaths) |>
  pivot_wider(names_from = "area", values_from = "pop") |>
  select(-age))

stan_data <- list(N = length(40:59),
                  M = 4,
                  K = 5,
                  y = y,
                  pop = pop,
                  z = q40,
                  age_c = (40:59)-40)
```

Fit the model:

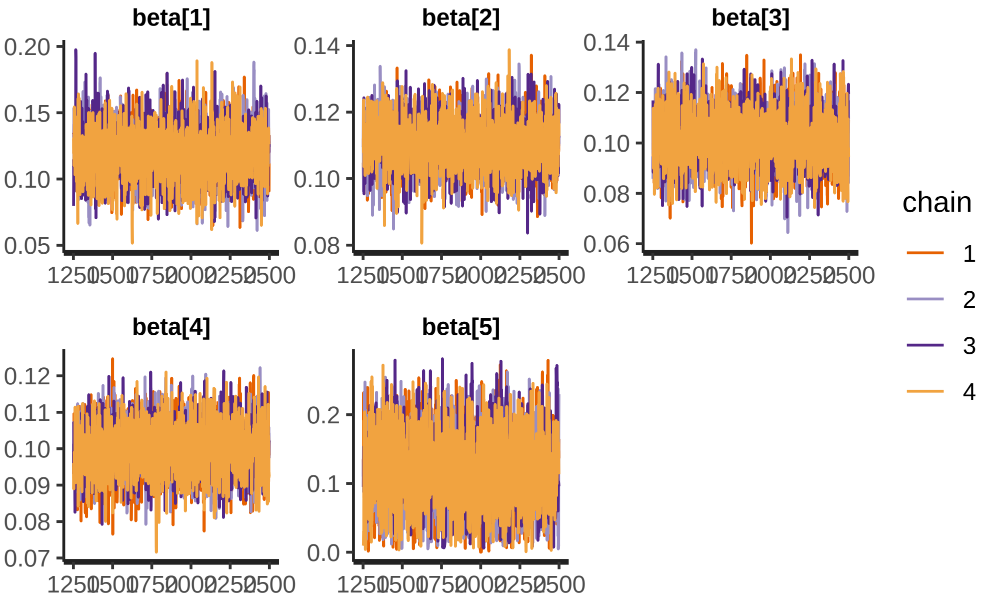
```
mod <- stan(file = "models/gomp_partial_data.stan",
            data = stan_data,
            seed = 15,
            control = list(adapt_delta = 0.96),
            iter = 2500,
            refresh = 0)
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDE
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Co
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHea
In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEige
/Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

```
traceplot(mod, c("log_alpha"))
```

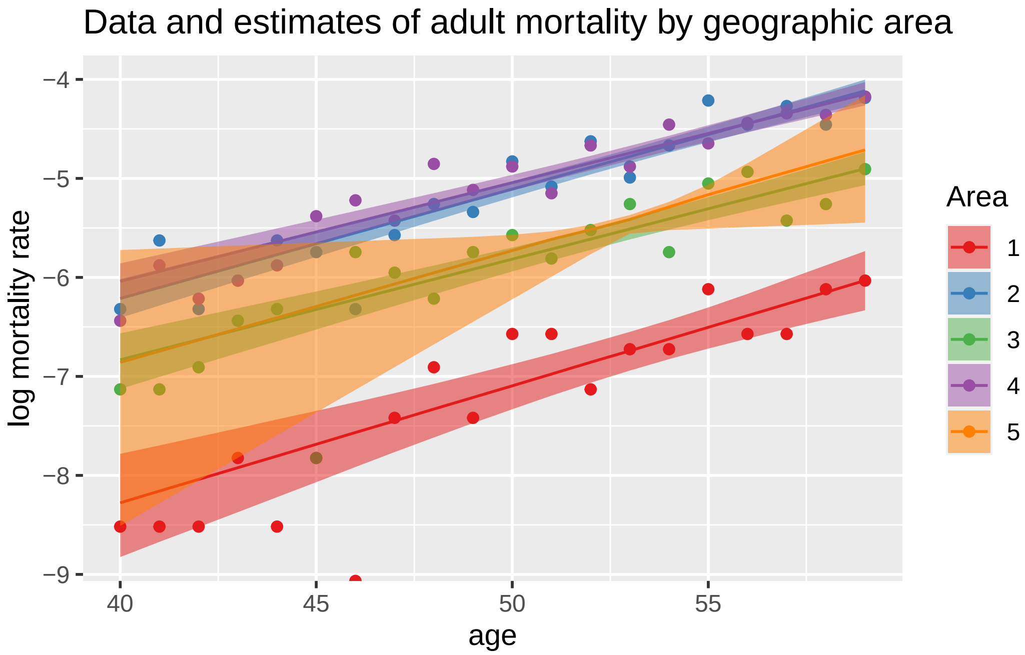


```
traceplot(mod, c("beta"))
```



Let's plot the fitted lines (with uncertainty) with the observed data for each region. Notice the difference in uncertainty around area 5!

```
mod |>
  gather_draws(mu[i,j]) |>
  median_qi() |>
  mutate(age = i-1+40) |>
  rename(area = j) |>
  left_join(df) |>
  mutate(log_mx = log(deaths/pop)) |>
  ggplot(aes(age, log_mx))+
  geom_point(aes(color = factor(area)))+
  geom_line(aes(age, .value, color = factor(area)))+
  geom_ribbon(aes(age, ymin = .lower, ymax = .upper, fill = factor(area)), alpha = 0.5)+
  scale_color_brewer(name = "Area", palette = "Set1")+
  scale_fill_brewer(name = "Area", palette = "Set1")+
  labs(title = "Data and estimates of adult mortality by geographic area", y = "log mortal
```



6.2 Compare parameter estimates against the truth

Here's what we estimated:

```
mod |>
  gather_draws(log_alpha[i], beta[i]) |>
  median_qi()
```

```
# A tibble: 10 x 8
       i .variable  .value  .lower .upper .width .point .interval
   <int> <chr>       <dbl>   <dbl>  <dbl>  <dbl> <chr>  <chr>
 1     1 beta       0.118   0.0823  0.158   0.95 median qi
 2     1 log_alpha -8.28   -8.82   -7.78    0.95 median qi
 3     2 beta       0.110   0.0959  0.125   0.95 median qi
 4     2 log_alpha -6.21   -6.41   -6.02    0.95 median qi
 5     3 beta       0.102   0.0821  0.123   0.95 median qi
 6     3 log_alpha -6.83   -7.13   -6.57    0.95 median qi
 7     4 beta       0.0995  0.0861  0.113   0.95 median qi
 8     4 log_alpha -6.04   -6.22   -5.86    0.95 median qi
 9     5 beta       0.113   0.0152  0.229   0.95 median qi
10     5 log_alpha -6.86   -8.51   -5.72    0.95 median qi
```

And here's the truth (the values underlying the simulation):

```
read_rds("data/true_params.rds")
```

```
# A tibble: 5 x 3
    are log_alpha   beta
  <int>     <dbl>  <dbl>
1     1     -7.98 0.0985
2     2     -6.38 0.125
3     3     -6.88 0.110
4     4     -5.93 0.0924
5     5     -7.48 0.154
```

6.3 Question for you

I arbitrarily set the population to be 5,000 people per age in the simulation. If we increased the population size to 50,000, what would you expect to happen to the simulated mortality rates? What about to the parameter estimates?