Bayesian Gompertz models for mortality

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. `rstan` 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 What is a Gompertz model?

In 1825 Benjamin Gompertz, an actuary in London, proposed the following two-parameter model for mortality:

$$\mu(x) = \alpha e^{\beta x}$$

where $\mu(x)$ is the instantaneous mortality rate (hazard) at age x. This model assumes that mortality increases exponentially with age, which is a pretty good assumption for adult mortality (above age 40 or so, with the exception of older ages).

4 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)
```

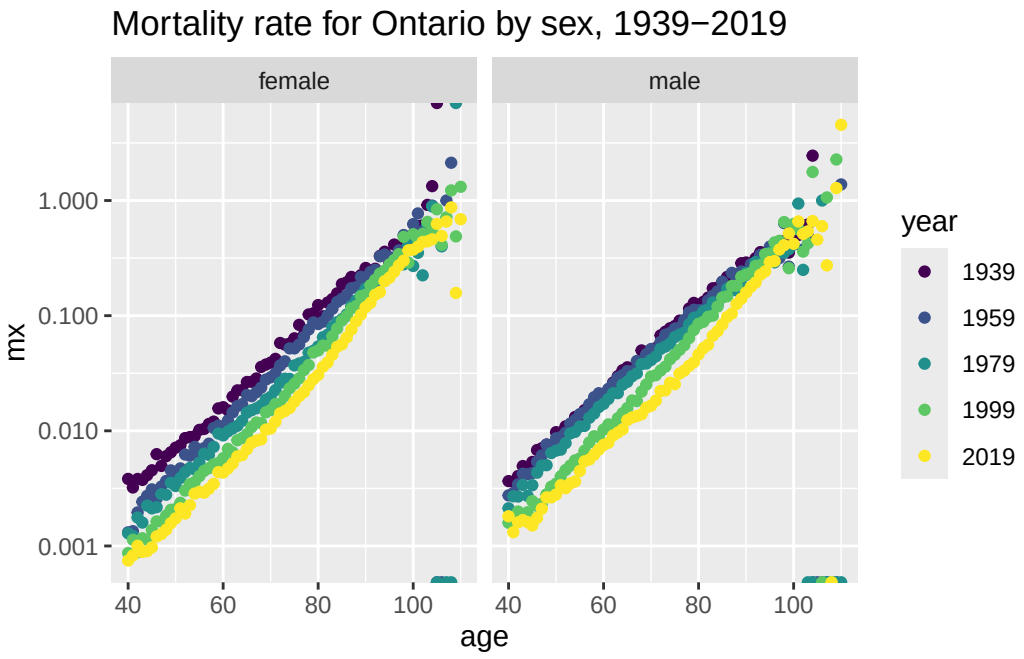
These data files are in 'wide' format. For our purposes it's going to be easier to work with in 'long format'. So let's do that and also clean some other stuff up:

```
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")
```



5 Base model

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

$$\mu(x) = \alpha e^{\beta x}$$

So we can write

$$\log \mu(x) = \log \alpha + \beta x$$

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):

```
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:

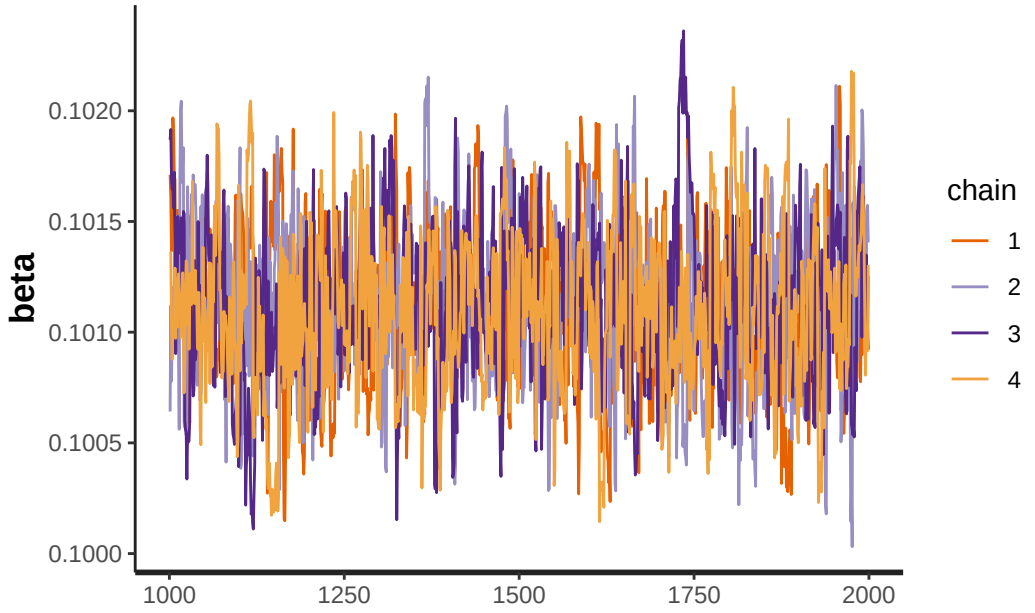
```
mod <- stan(file = "../models/gomp.stan",
            data = stan_data,
            seed = 123,
            refresh = 0)
```

```
names(mod)
```

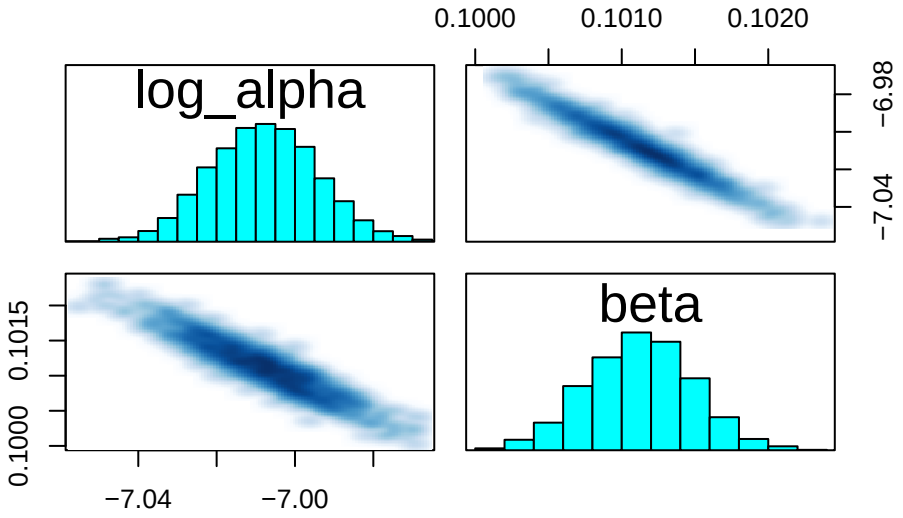
```
 [1] "log_alpha" "beta"      "mu[1]"     "mu[2]"     "mu[3]"     "mu[4]"
 [7] "mu[5]"     "mu[6]"     "mu[7]"     "mu[8]"     "mu[9]"     "mu[10]"
[13] "mu[11]"    "mu[12]"    "mu[13]"    "mu[14]"    "mu[15]"    "mu[16]"
[19] "mu[17]"    "mu[18]"    "mu[19]"    "mu[20]"    "mu[21]"    "mu[22]"
[25] "mu[23]"    "mu[24]"    "mu[25]"    "mu[26]"    "mu[27]"    "mu[28]"
[31] "mu[29]"    "mu[30]"    "mu[31]"    "mu[32]"    "mu[33]"    "mu[34]"
[37] "mu[35]"    "mu[36]"    "mu[37]"    "mu[38]"    "mu[39]"    "mu[40]"
[43] "mu[41]"    "mu[42]"    "mu[43]"    "mu[44]"    "mu[45]"    "mu[46]"
[49] "mu[47]"    "mu[48]"    "mu[49]"    "mu[50]"    "mu[51]"    "mu[52]"
[55] "mu[53]"    "mu[54]"    "mu[55]"    "mu[56]"    "mu[57]"    "mu[58]"
[61] "mu[59]"    "mu[60]"    "mu[61]"    "mu[62]"    "mu[63]"    "mu[64]"
[67] "mu[65]"    "lp__"
```

Some quick model checks, looking at the traceplot and pairwise densities

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




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



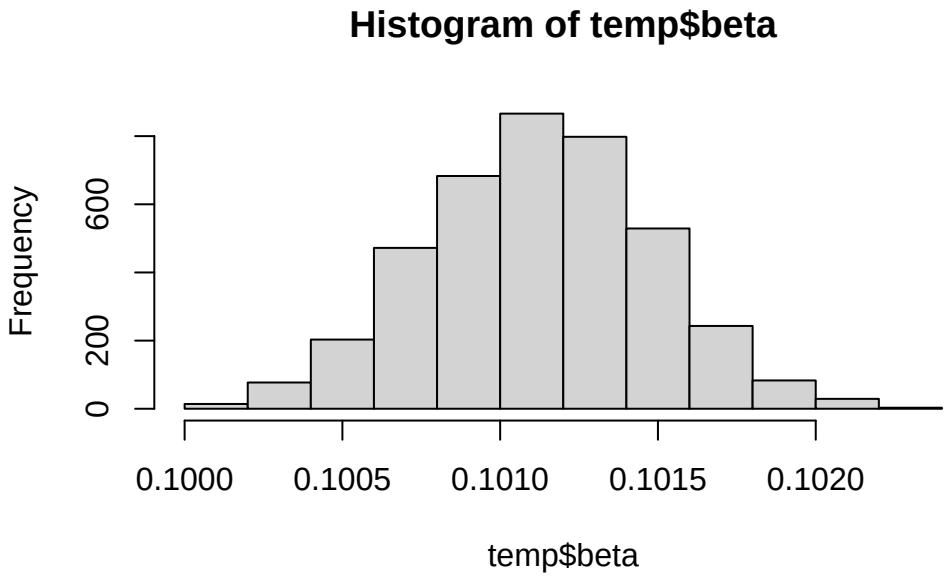

```
temp <- rstan::extract(mod)
names(temp)
```

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

```
median(temp$beta)
```

```
[1] 0.1011281
```

```
hist(temp$beta)
```



5.1 Extract the parameter estimates using `tidyverse`

```
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  -7.03  -6.98    0.95 median qi
```

5.2 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)
```

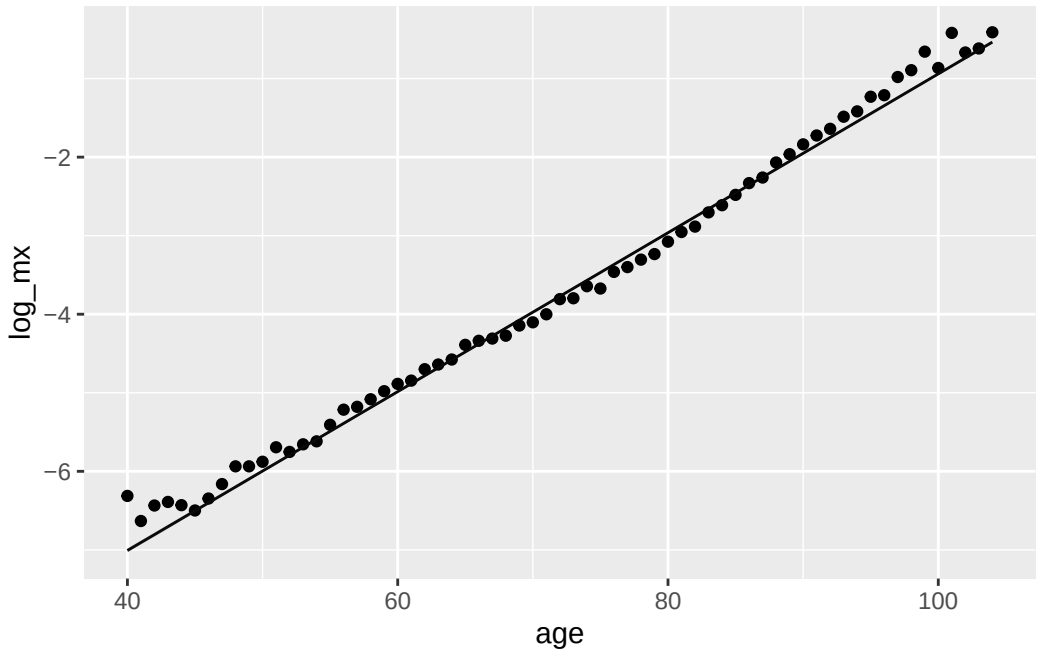
```
Model Info:
 function:     stan_glm
 family:       poisson [log]
 formula:      deaths ~ age_c + offset(log(pop))
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 65
 predictors:   2

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 808.7    5.0 802.4 808.7 815.1

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  1991
age_c         0.0  1.0  2240
mean_PPD      0.1  1.0  2777
log-posterior 0.0  1.0  1727

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

5.3 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)
```



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()
```

```
  log_alpha log_alpha.lower log_alpha.upper      beta beta.lower beta.upper
1 -7.007979       -7.034161       -6.981138 0.1011281  0.1004227  0.1018241
  mode_age mode_age.lower mode_age.upper .width .point .interval
1 46.64084       46.55731       46.72334   0.95 median        qi
```

6 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:

```
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)
```

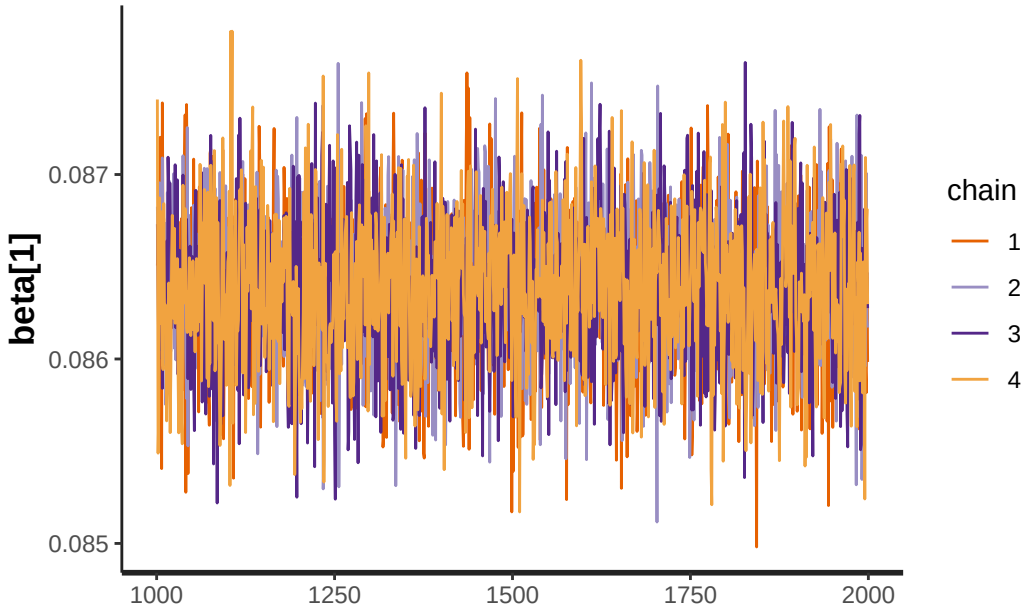
```
summary(mod)$summary[paste0("beta[", 1:(length(years)), "]"),]
```

```
              mean      se_mean           sd       2.5%        25%        50%
beta[1]  0.08637862 5.615217e-06 0.0003982933 0.08560144 0.08611433 0.08637399
beta[2]  0.08610158 5.442197e-06 0.0003684184 0.08539399 0.08585290 0.08610017
beta[3]  0.08651788 5.414198e-06 0.0003510223 0.08580702 0.08628288 0.08652359
beta[4]  0.08682847 4.694212e-06 0.0003542371 0.08612976 0.08659631 0.08682399
beta[5]  0.08740523 5.061156e-06 0.0003470550 0.08671656 0.08717520 0.08740117
beta[6]  0.08748393 5.005646e-06 0.0003519533 0.08680129 0.08724565 0.08748208
beta[7]  0.08775353 4.740365e-06 0.0003557757 0.08706661 0.08750971 0.08775476
beta[8]  0.08775550 4.913209e-06 0.0003525789 0.08707942 0.08751248 0.08774739
beta[9]  0.08714405 5.018883e-06 0.0003489820 0.08645937 0.08691100 0.08714367
beta[10] 0.08750373 5.069036e-06 0.0003431962 0.08681338 0.08727461 0.08750755
beta[11] 0.08849780 4.976968e-06 0.0003665736 0.08776053 0.08825227 0.08849827
beta[12] 0.08984495 5.261571e-06 0.0003499203 0.08917015 0.08960616 0.08984220
beta[13] 0.09052012 4.669489e-06 0.0003544987 0.08983324 0.09027365 0.09052286
beta[14] 0.09139881 4.727145e-06 0.0003552288 0.09071134 0.09116002 0.09140032
beta[15] 0.09214967 5.278387e-06 0.0003415211 0.09146699 0.09192709 0.09214568
beta[16] 0.09235668 4.642073e-06 0.0003435534 0.09170281 0.09212583 0.09235461
beta[17] 0.09317898 4.806490e-06 0.0003422522 0.09250904 0.09295665 0.09317895
beta[18] 0.09361941 4.755997e-06 0.0003543166 0.09292795 0.09337718 0.09361663
beta[19] 0.09449054 4.830643e-06 0.0003388760 0.09384161 0.09425834 0.09448656
beta[20] 0.09590685 4.817062e-06 0.0003403181 0.09523212 0.09567694 0.09590492
beta[21] 0.09572516 5.205427e-06 0.0003463620 0.09505124 0.09549059 0.09572375
beta[22] 0.09560717 4.798968e-06 0.0003377937 0.09494854 0.09537865 0.09560536
beta[23] 0.09666523 5.173136e-06 0.0003388139 0.09601247 0.09644240 0.09666453
beta[24] 0.09687333 5.016186e-06 0.0003378535 0.09620225 0.09665068 0.09687717
beta[25] 0.09708172 5.075969e-06 0.0003348943 0.09640940 0.09685690 0.09708159
beta[26] 0.09803235 4.346514e-06 0.0003315032 0.09737925 0.09781175 0.09803340
beta[27] 0.09873404 4.792988e-06 0.0003343830 0.09809434 0.09850857 0.09874058
beta[28] 0.09982462 4.804610e-06 0.0003305147 0.09917353 0.09959536 0.09982823
beta[29] 0.10070598 4.606149e-06 0.0003359334 0.10004291 0.10047530 0.10070647
beta[30] 0.10202393 4.811685e-06 0.0003271481 0.10141693 0.10180444 0.10201660
beta[31] 0.10279522 4.946709e-06 0.0003325599 0.10213722 0.10257197 0.10279194
beta[32] 0.10218898 4.684997e-06 0.0003302130 0.10155613 0.10196276 0.10219361
beta[33] 0.10154142 4.913614e-06 0.0003379078 0.10088382 0.10130944 0.10154415
beta[34] 0.10109330 4.559001e-06 0.0003269494 0.10043578 0.10087444 0.10109538
beta[35] 0.10135396 4.243459e-06 0.0003145798 0.10073021 0.10114373 0.10135697
beta[36] 0.10127878 4.195499e-06 0.0003226281 0.10064017 0.10106426 0.10128002
beta[37] 0.10169618 4.428464e-06 0.0003254565 0.10105994 0.10148285 0.10168986
beta[38] 0.10160432 4.541919e-06 0.0003186363 0.10097502 0.10138417 0.10160854
beta[39] 0.10157442 4.690420e-06 0.0003204719 0.10095175 0.10135536 0.10157498
```

```
beta[40] 0.10205214 4.391470e-06 0.0003136110 0.10143793 0.10183975 0.10205770
beta[41] 0.10200972 4.570801e-06 0.0003269815 0.10136915 0.10178795 0.10201694
beta[42] 0.10191818 4.424411e-06 0.0003084719 0.10130880 0.10171255 0.10191854
beta[43] 0.10195768 4.216013e-06 0.0003159277 0.10132049 0.10174920 0.10196383
beta[44] 0.10183430 4.436310e-06 0.0003120869 0.10123466 0.10161956 0.10182447
beta[45] 0.10212274 4.311270e-06 0.0003057219 0.10151945 0.10191839 0.10212256
beta[46] 0.10228937 4.251937e-06 0.0002960884 0.10170069 0.10209173 0.10229349
beta[47] 0.10218294 4.491797e-06 0.0003062637 0.10158407 0.10197511 0.10218712
beta[48] 0.10136531 4.148454e-06 0.0003030559 0.10077495 0.10116798 0.10136980
beta[49] 0.10180385 4.124885e-06 0.0003015646 0.10121663 0.10160552 0.10180487
beta[50] 0.10137931 4.056217e-06 0.0002986941 0.10079408 0.10117289 0.10138585
beta[51] 0.10076087 3.873514e-06 0.0002942502 0.10017702 0.10056703 0.10075913
                75%      97.5%    n_eff      Rhat
beta[1]  0.08664917 0.08714652 5031.216 0.9999631
beta[2]  0.08635031 0.08682391 4582.834 0.9992589
beta[3]  0.08674821 0.08720761 4203.403 0.9993241
beta[4]  0.08706509 0.08750969 5694.593 0.9992659
beta[5]  0.08763562 0.08809308 4702.156 1.0006264
beta[6]  0.08771983 0.08817228 4943.675 1.0005131
beta[7]  0.08799053 0.08845484 5632.860 0.9993100
beta[8]  0.08799780 0.08844597 5149.703 0.9995369
beta[9]  0.08738526 0.08783731 4834.949 0.9999654
beta[10] 0.08773256 0.08816442 4583.890 0.9998631
beta[11] 0.08875450 0.08917748 5424.909 0.9992249
beta[12] 0.09008766 0.09053783 4422.903 0.9999262
beta[13] 0.09076341 0.09122149 5763.557 1.0002520
beta[14] 0.09162902 0.09211342 5647.010 1.0000062
beta[15] 0.09237619 0.09283166 4186.322 0.9995712
beta[16] 0.09258478 0.09304118 5477.275 0.9995991
beta[17] 0.09340879 0.09384387 5070.333 0.9995250
beta[18] 0.09386584 0.09430520 5550.087 0.9993393
beta[19] 0.09471768 0.09517322 4921.210 0.9996376
beta[20] 0.09613470 0.09657807 4991.208 0.9995487
beta[21] 0.09595790 0.09641714 4427.391 1.0000775
beta[22] 0.09584482 0.09625143 4954.585 1.0005885
beta[23] 0.09688681 0.09732697 4289.580 1.0005519
beta[24] 0.09709658 0.09753264 4536.381 0.9995273
beta[25] 0.09730428 0.09773190 4352.889 0.9994075
beta[26] 0.09825651 0.09866986 5816.923 1.0004901
beta[27] 0.09895754 0.09939850 4867.161 1.0016361
beta[28] 0.10004394 0.10048700 4732.224 1.0001095
beta[29] 0.10094029 0.10135766 5319.003 0.9992013
beta[30] 0.10223621 0.10267131 4622.687 0.9997668
```

```
beta[31] 0.10301571 0.10345067 4519.673 0.9995930
beta[32] 0.10240463 0.10284404 4967.863 0.9994771
beta[33] 0.10176793 0.10220526 4729.274 0.9996908
beta[34] 0.10131226 0.10173301 5143.062 0.9995115
beta[35] 0.10156629 0.10196587 5495.682 1.0001080
beta[36] 0.10149778 0.10190272 5913.399 0.9995192
beta[37] 0.10192095 0.10232662 5401.068 0.9994758
beta[38] 0.10181806 0.10224378 4921.661 0.9995370
beta[39] 0.10179195 0.10220123 4668.276 0.9994392
beta[40] 0.10226711 0.10266331 5099.912 0.9997574
beta[41] 0.10222658 0.10264495 5117.546 0.9994766
beta[42] 0.10212445 0.10251034 4860.942 0.9996196
beta[43] 0.10216269 0.10259467 5615.282 1.0000650
beta[44] 0.10204339 0.10245796 4948.883 1.0001084
beta[45] 0.10232604 0.10270647 5028.550 1.0001978
beta[46] 0.10248761 0.10287575 4849.189 1.0005840
beta[47] 0.10238704 0.10277553 4648.906 0.9991086
beta[48] 0.10157322 0.10195125 5336.701 0.9995298
beta[49] 0.10200377 0.10241356 5344.869 0.9993926
beta[50] 0.10158624 0.10194382 5422.640 0.9991970
beta[51] 0.10096056 0.10134748 5770.631 0.9992983
```

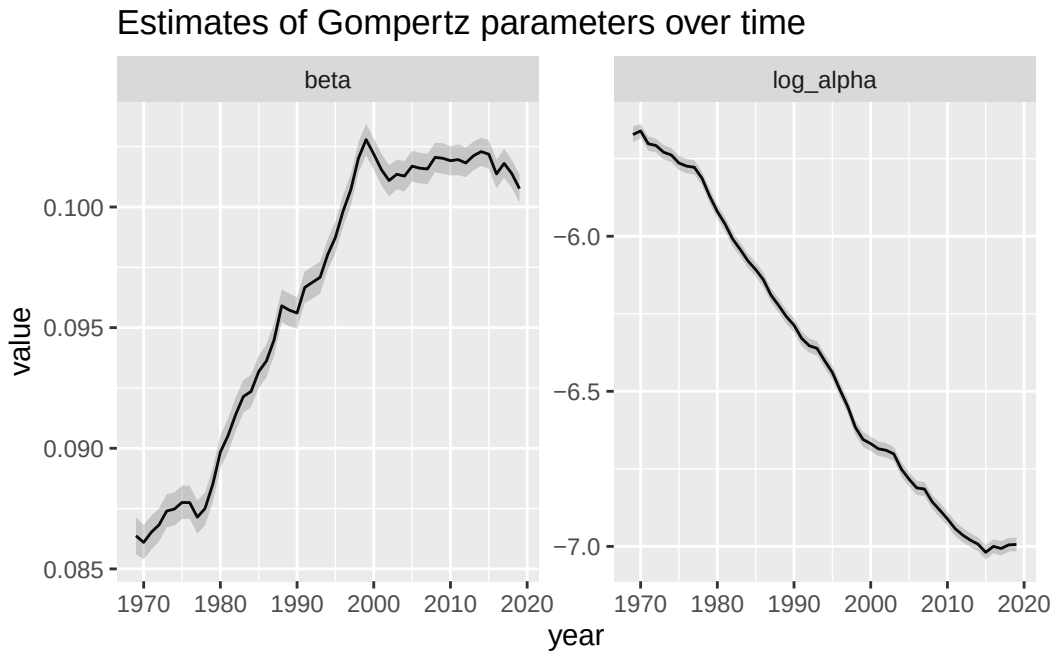
```
traceplot(mod, pars = c("beta[1]"))
```



6.1 Plot parameter estimates over time

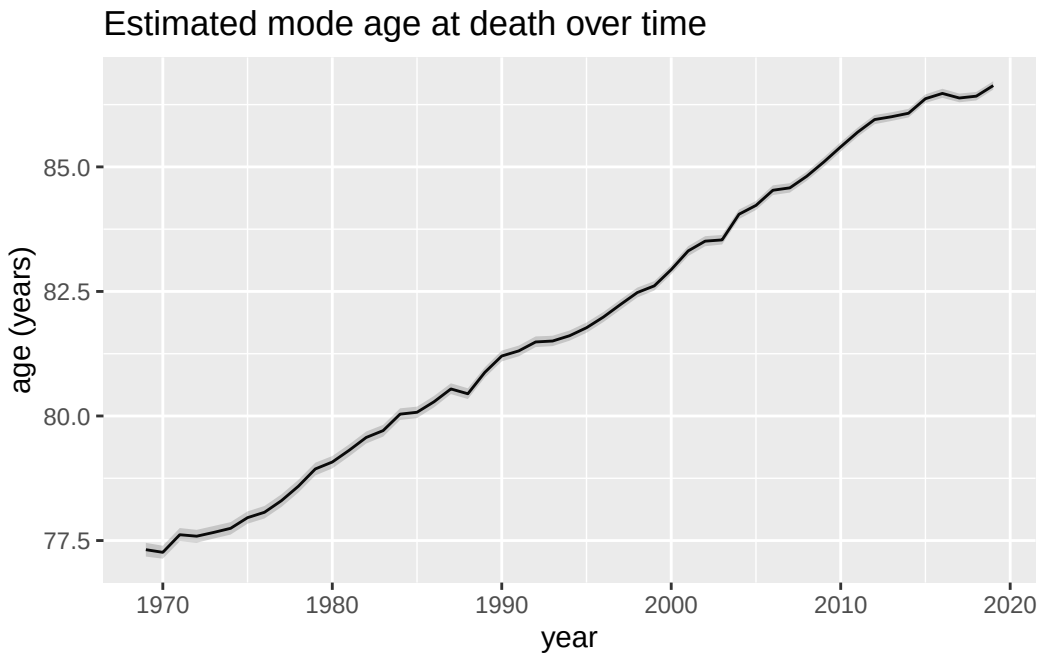
Now we can plot the parameter estimates over time:

```
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)")
```



6.2 Question for you

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

7 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
```

7.1 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:

$$ {}_1p_x = e^{-\mu_x} $$

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

$$ {}_{20}q_{40} = 1 - \prod_{x=40}^{60} {}_1p_x $$

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

7.2 Fit the model

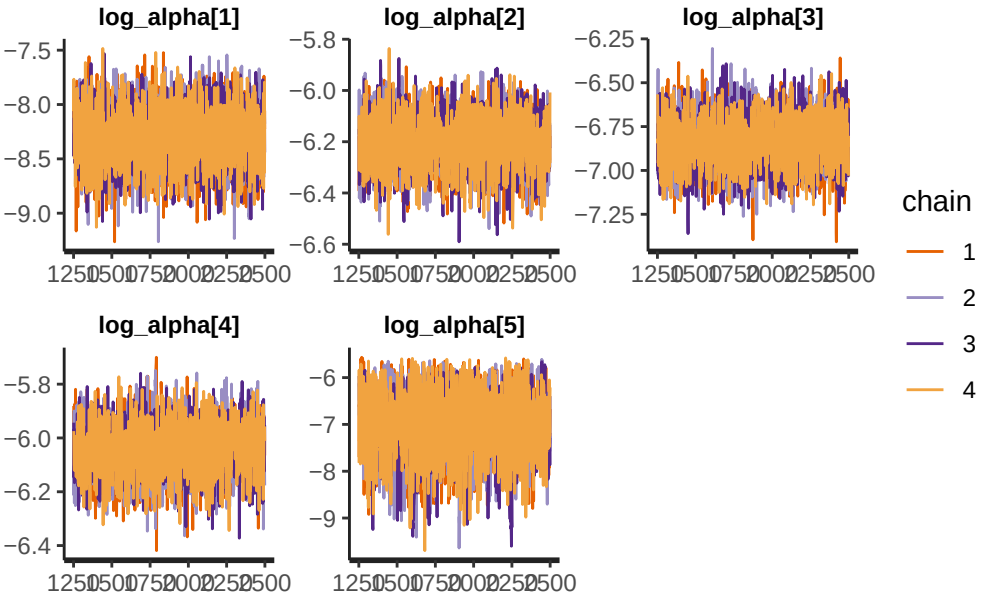
Get the data in the right format. Note that everything is a matrix now because we have more than one area:

```
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)
```

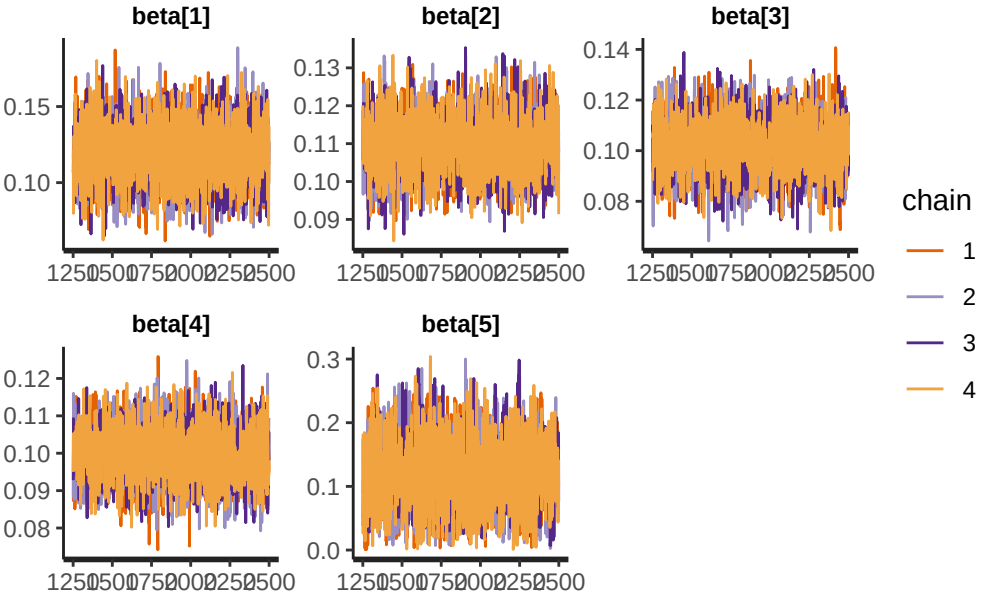
Fit the model:

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

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



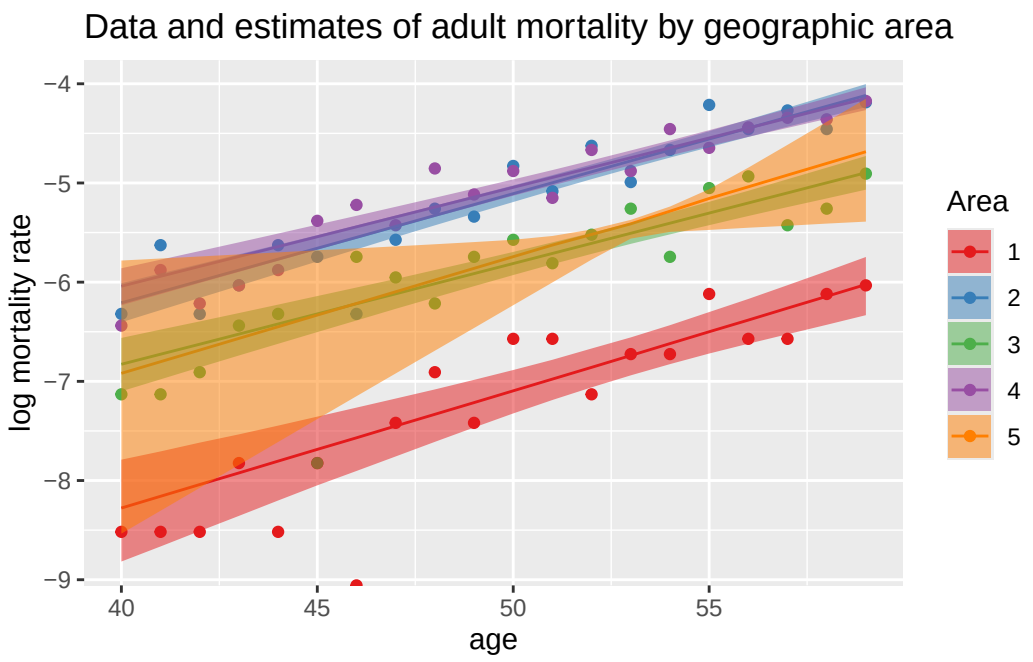

```
traceplot(mod, c("beta"))
```



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 mortality
```



7.3 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   0.0831  0.156   0.95 median qi
 2     1 log_alpha -8.28   -8.82   -7.79    0.95 median qi
 3     2 beta       0.110   0.0957  0.125   0.95 median qi
 4     2 log_alpha -6.21   -6.40   -6.01    0.95 median qi
 5     3 beta       0.102   0.0815  0.121   0.95 median qi
 6     3 log_alpha -6.83   -7.10   -6.56    0.95 median qi
 7     4 beta       0.0996  0.0860  0.113   0.95 median qi
 8     4 log_alpha -6.04   -6.23   -5.86    0.95 median qi
 9     5 beta       0.118   0.0205  0.231   0.95 median qi
10     5 log_alpha -6.92   -8.54   -5.78    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
```