Multilevel models, part 1

Frank Edwards

4/17/2020

Multilevel data

When your data has structure

Many natural and social processes have inherent groupings

- Organisms live within ecosystems

When your data has structure

Many natural and social processes have inherent groupings

- Organisms live within ecosystems
- People live within neighborhoods (within cities)

When your data has structure

Many natural and social processes have inherent groupings

- Organisms live within ecosystems
- People live within neighborhoods (within cities)
- Students learn within classrooms (within schools, within districts, ...)

When your data has structure

Many natural and social processes have inherent groupings

- Organisms live within ecosystems
- People live within neighborhoods (within cities)
- Students learn within classrooms (within schools, within districts, ...)

Structured data are often clustered

These groups often lead to patterns in our data called *clustering*

- Variables are often correlated within clusters
- These correlations can be due observed variables
- But they can also be due to unobserved (or unobservable) features of each cluster
- If we don't account for clustering, our inferences will be misleading

Gapminder data

Let's look at these data on variation in life expectancy, population, and per capita GDP across countries over time

```
### install.packages("gapminder")
library(gapminder)
gapminder %>% sample_n(5)
```

```
## # A tibble: 5 x 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Montenegro Europe     1992    75.4   621621     7003.
## 2 Hungary    Europe     1982    69.4 10705535    12546.
## 3 Benin      Africa     1992    53.9  4981671     1191.
## 4 Malawi     Africa     1977    43.8  5637246      663.
## 5 Thailand   Asia       1992    67.3 56667095     4617.
```

A life expectancy model

Say we want to learn something about average mortality rates within a country. We could fit a model for life expectancy L.

$$L_i \sim Normal(\mu, \sigma)$$

$$\mu_i = \alpha$$

$$\alpha \sim Normal(55, 15)$$

$$sigma \sim Exponential(0.5)$$

Estimating the model

This model, with no included parameters for group differences, is called a *complete pooling* model. Data from all groups is used to estimate a single (set) of global parameters.

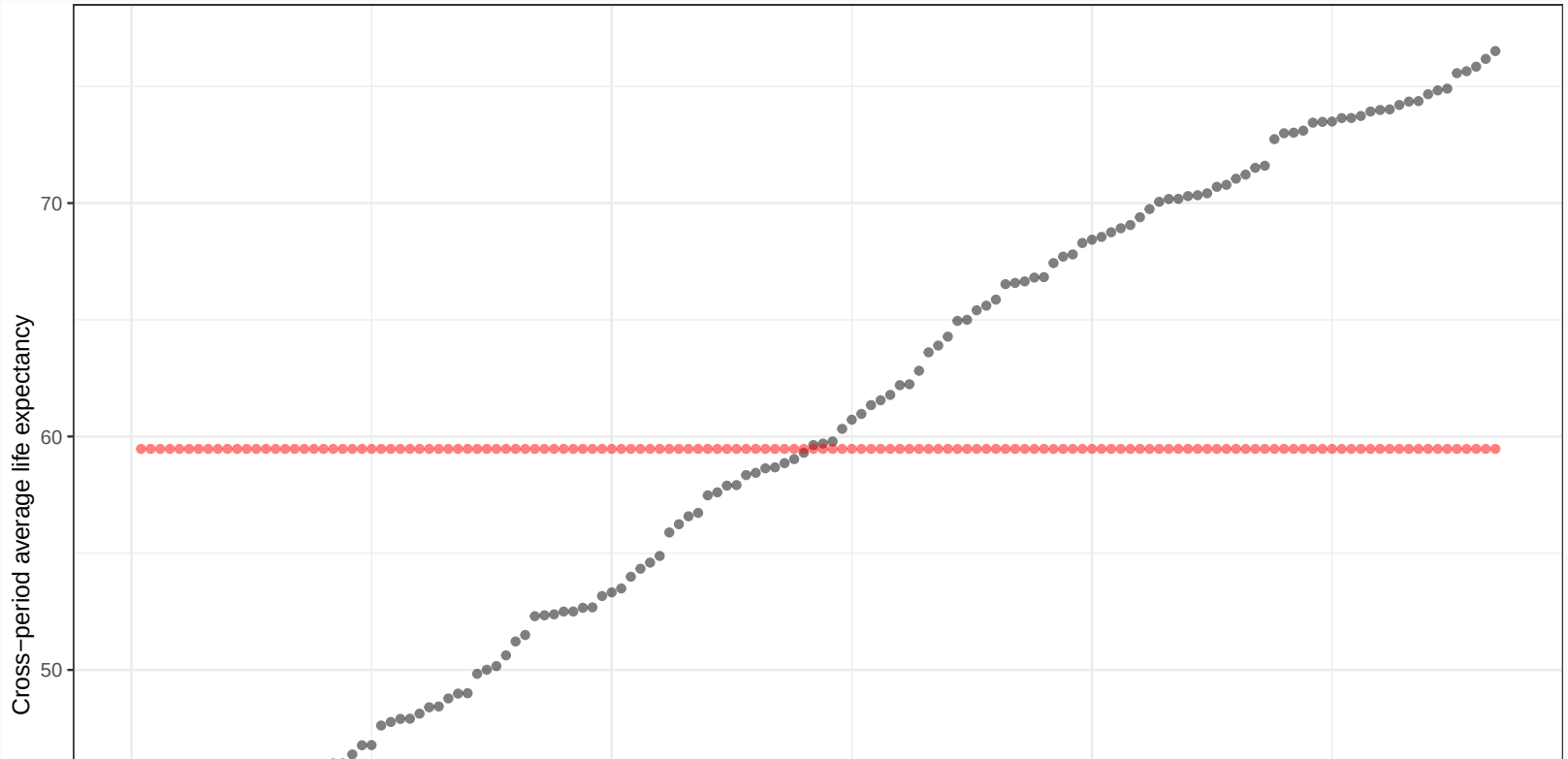
```
dat<-gapminder %>%
  mutate(L = lifeExp,
         country = factor(country)) %>%
  select(L, country)

m_LE0<-ulam(alist(
  L ~ dnorm(mu, sigma),
  mu <- a,
  a ~ dnorm(55, 15),
  sigma ~ dexp(0.5)),
  data = dat, chains = 4, cores = 4, log_lik = T)
```

Visualizing the model

```
sim_dat <- data.frame(country = unique(dat$country))
### draw posterior predictions of L for each country
sims <- link(m_LE0, sim_dat)
sims_PI<-apply(sims, 2, PI)
### compute empirical means, attach posterior estimates of \alpha[country]
plot_dat<-dat %>%
  group_by(country) %>%
  summarise(L_bar = mean(L)) %>%
  ungroup() %>%
  mutate(a_bar = apply(sims, 2, mean),
         a_lwr = sims_PI[1,],
         a_upr = sims_PI[2,]) %>%
  arrange(L_bar)
```

Visualizing the model

```
ggplot(plot_dat,
       aes(x = 1:nrow(plot_dat),
           y = L_bar)) +
  geom_point(alpha = 0.5) +
  geom_point(aes(y = a_bar), color = "red", alpha = 0.5) +
  labs(x = "", y = "Cross-period average life expectancy")
```



Weird - what happened?

- An *intercept is a mean*
- Expected values in a complete pooling intercept-only model will be approximately equal to the empirical mean of the data

```
precis(m_LE0)
```

```
##             mean         sd     5.5%    94.5%    n_eff    Rhat4
## a     59.46917 0.3127659 58.94772 59.99202 1480.212 1.001063
## sigma 12.91074 0.2160624 12.56487 13.25789 1342.033 1.001520
```

```
mean(dat$L)
```

```
## [1] 59.47444
```

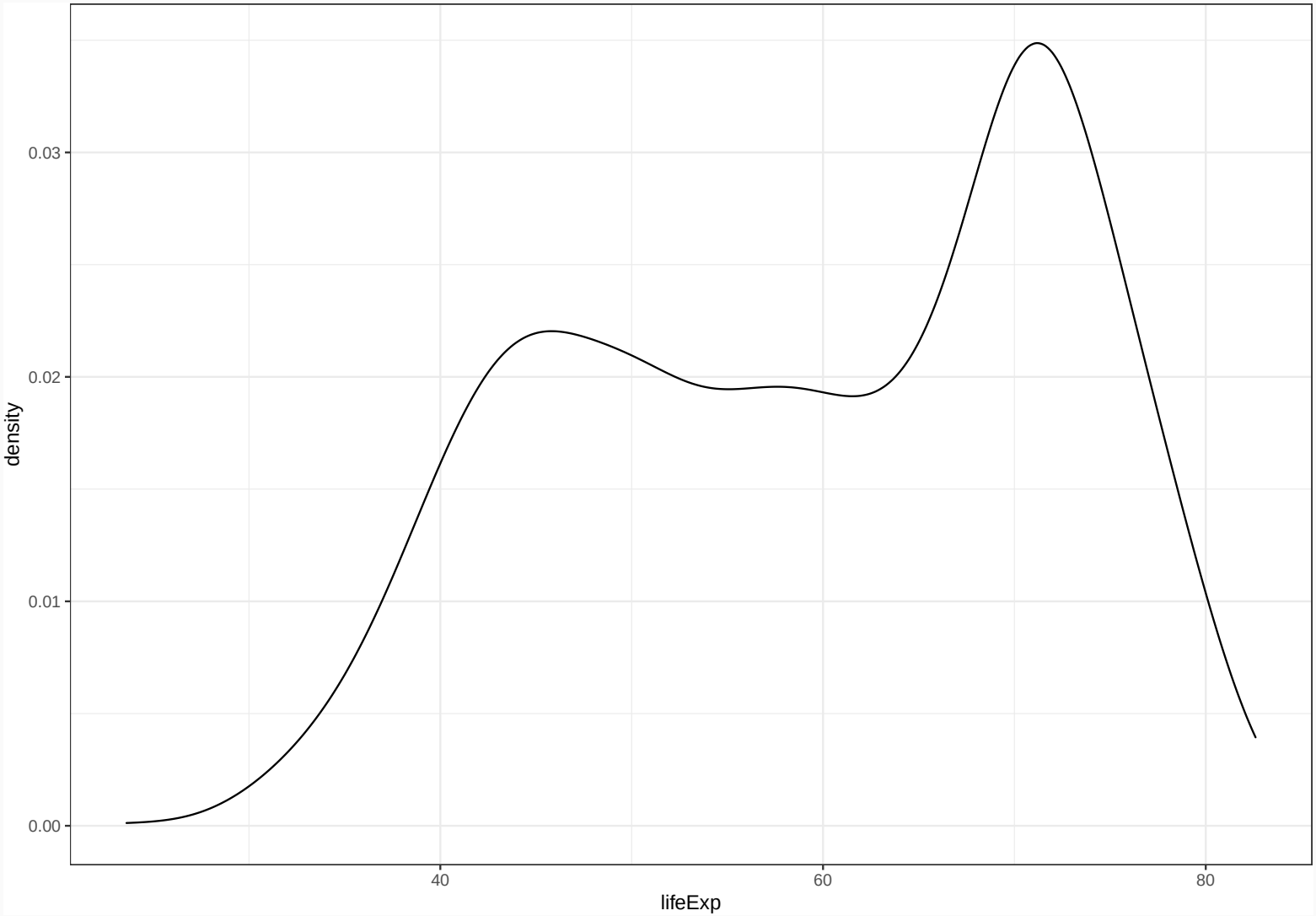
When to avoid complete pooling

Sometimes units in the data have features that make them similar to other units.

- Repeated observations of countries over time (longitudinal or time series data)
- Observations of many countries from the same time-period (cross-sectional data)
- Countries are nested within larger geographies (regions, continents, regions)
- We can adjust our models to learn from these features and improve inference

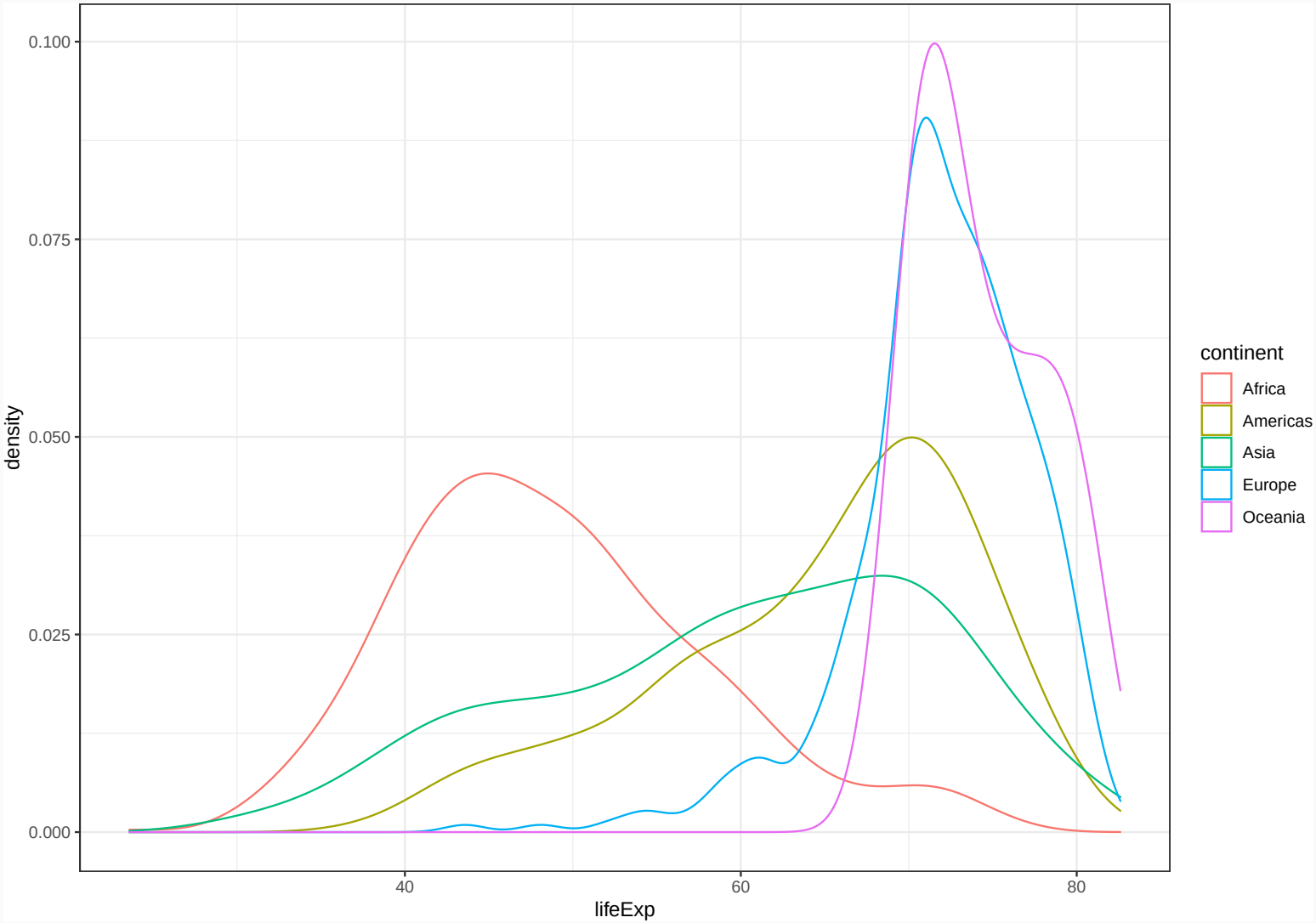
Life expectancy in the data: complete pooling

```
ggplot(gapminder, aes(x = lifeExp)) +
  geom_density()
```



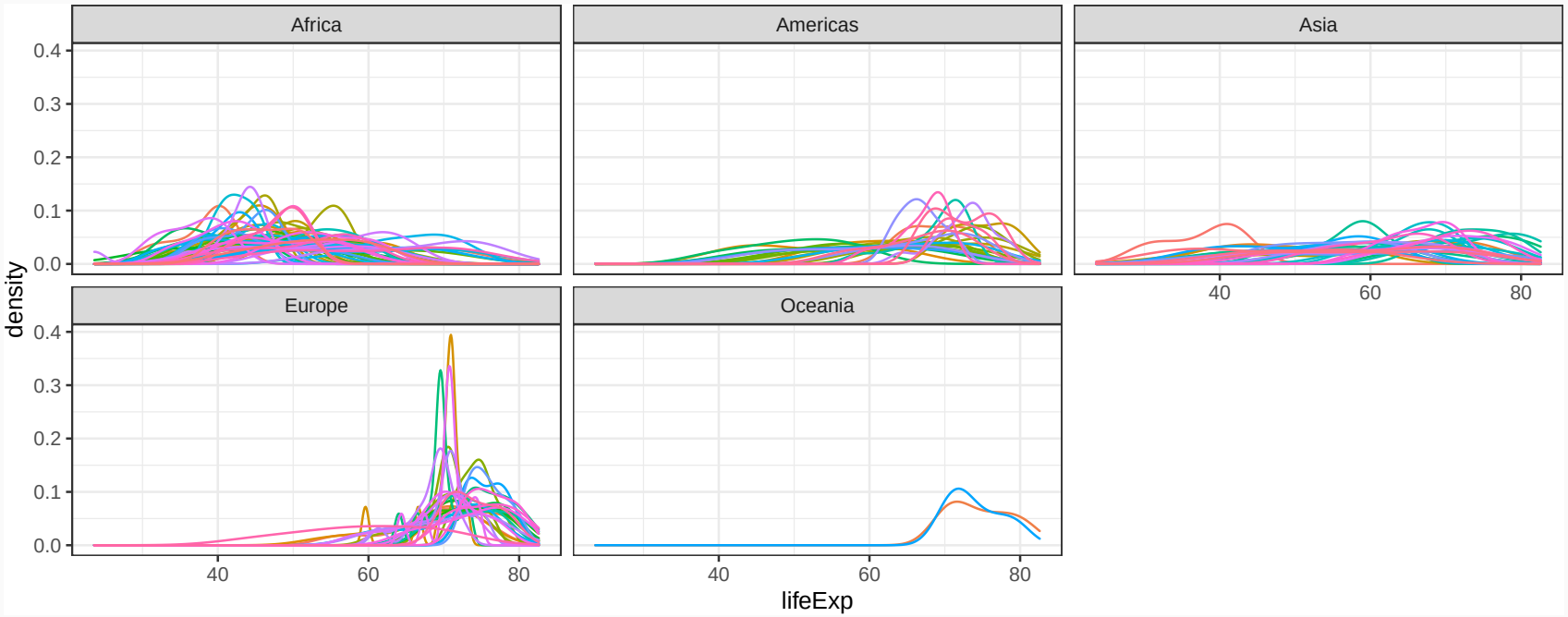
One source of clustering: geography

```
ggplot(gapminder, aes(x = lifeExp, color = continent)) +
  geom_density()
```



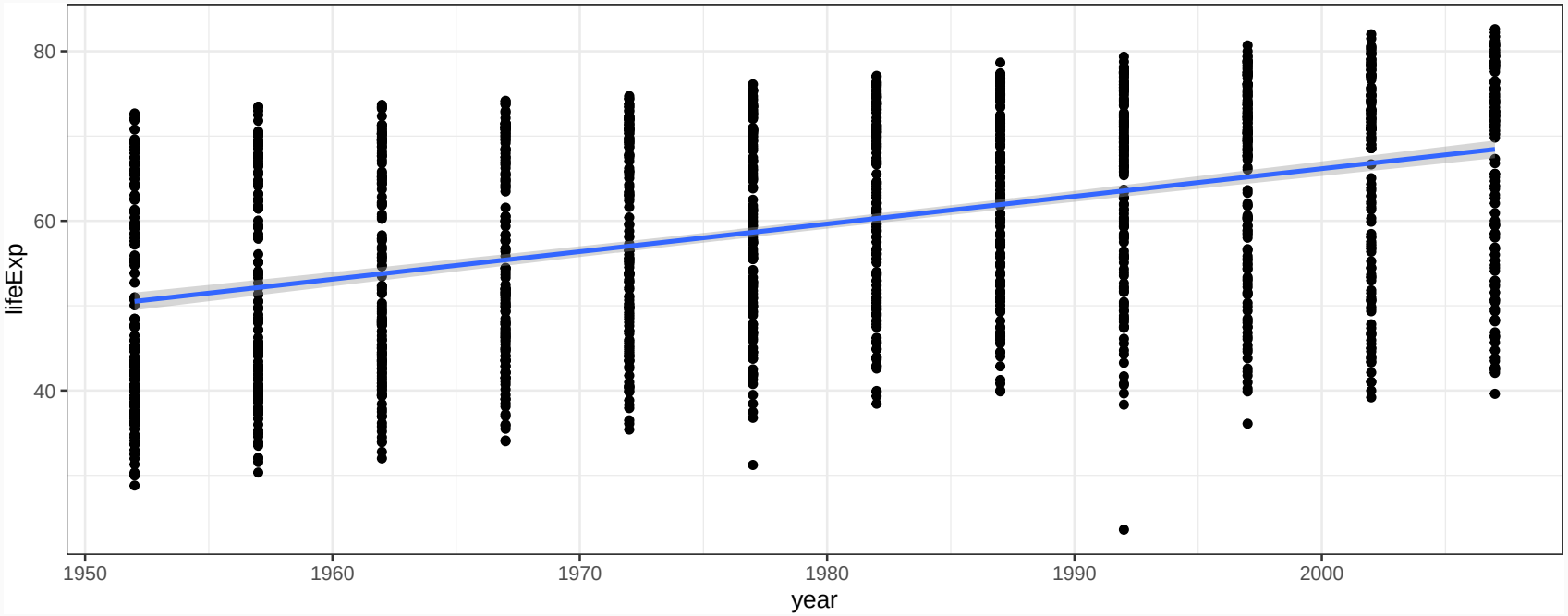
More geographic clustering

```
ggplot(gapminder, aes(x = lifeExp, color = country)) +
  geom_density() +
  facet_wrap(~continent) +
  guides(color = F)
```



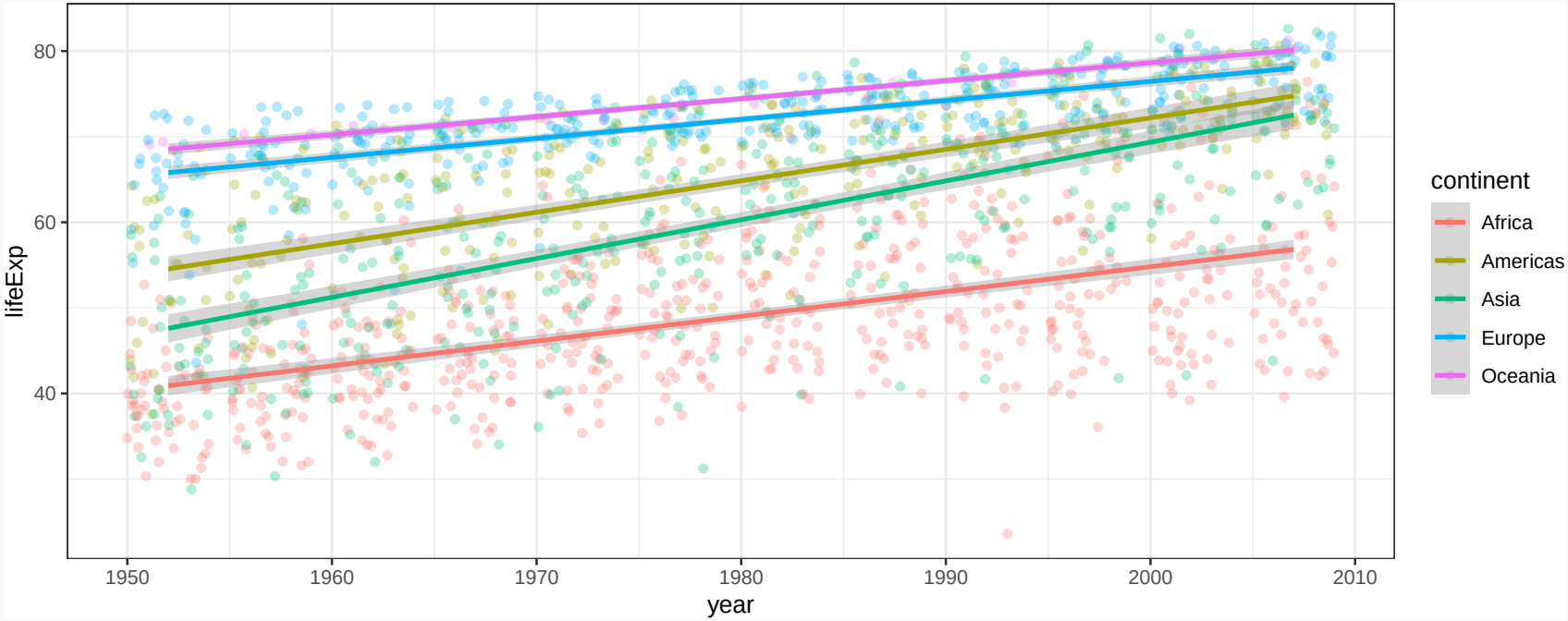
Temporal clustering

```
ggplot(gapminder, aes(y = lifeExp, x = year)) +
  geom_point() +
  geom_smooth(method = "lm")
```



Temporal and geographic clustering

```
ggplot(gapminder, aes(y = lifeExp, x = year,
                      color = continent)) +
  geom_jitter(alpha = 0.3) +
  geom_smooth(method = "lm")
```



Intercepts as regression parameters

Intercepts as group means

Say we want to learn something about average mortality rates within a country. We could fit a model for life expectancy L with an *intercept* for each.

$$L_i \sim Normal(\mu, \sigma)$$

$$\mu_i = \alpha[country]_i$$

$$\alpha[country] \sim Normal(55, 15)$$

$$sigma \sim Exponential(0.5)$$

Estimating the country intercept model

```
dat<-gapminder %>%
  mutate(L = lifeExp,
         country = factor(country)) %>%
  select(L, country)

m_LE1<-ulam(alist(
  L ~ dnorm(mu, sigma),
  mu <- a[country],
  a[country] ~ dnorm(55, 15),
  sigma ~ dexp(0.5)),
  data = dat, chains = 4, cores = 4, log_lik = T)
```

This isn't fun to look at, but here are the parameter estimates

```
precis(m_LE1, depth = 2)
```

```
##             mean        sd      5.5%     94.5%    n_eff     Rhat4
## a[1]   37.794723 1.9552488 34.561622 40.869483 2425.458 0.9981805
## a[2]   68.139322 1.9697552 64.986093 71.353148 2911.147 0.9997227
## a[3]   58.878189 1.9463504 55.843139 62.065252 2668.539 0.9989501
## a[4]   38.236257 1.9502431 35.193604 41.319720 2790.315 0.9990207
## a[5]   68.784599 2.0034685 65.499311 71.963657 2910.434 0.9985379
## a[6]   74.265192 1.9803696 71.125037 77.439935 2437.720 0.9994787
## a[7]   72.827750 1.9409308 69.634266 75.951968 2850.888 0.9992220
## a[8]   65.444307 1.9658548 62.227891 68.536831 3858.634 0.9998644
## a[9]   49.947797 1.9971089 46.737431 52.984535 3465.074 0.9988677
## a[10]  73.286899 1.9769894 70.040675 76.422413 3033.728 1.0002747
## a[11]  48.855847 1.9235581 45.736595 51.926087 2183.197 0.9983192
## a[12]  52.587467 1.9695790 49.352761 55.681736 4402.191 0.9984315
## a[13]  67.477156 1.9546670 64.389977 70.666631 3201.706 0.9989854
## a[14]  54.588256 1.8773658 51.605896 57.550517 2634.037 0.9998434
## a[15]  62.149316 1.9512479 59.064172 65.219276 3005.988 0.9997484
## a[16]  69.422167 1.9931042 66.249306 72.615142 2422.450 0.9990170
## a[17]  44.921471 2.0598373 41.464423 48.283062 2765.144 0.9987852
## a[18]  44.948485 2.1037297 41.599817 48.189509 3157.583 0.9999519
## a[19]  48.028818 2.0122873 44.830580 51.160900 3044.184 0.9999533
## a[20]  48.215530 1.9443329 45.040480 51.302376 2444.746 0.9990070
## a[21]  74.541218 1.9332401 71.510398 77.664475 2677.004 0.9991570
## a[22]  44.065459 1.9788100 40.861183 47.204496 2720.978 0.9992590
## a[23]  46.894332 1.9465588 43.701807 50.045771 2522.182 0.9995021
## a[24]  67.132980 2.0027722 63.959645 70.283215 3688.407 0.9982936
## a[25]  61.618570 1.9221127 58.637406 64.807806 3153.562 0.9993603
## a[26]  63.727650 1.9670829 60.551928 66.834272 3318.989 0.9993681
```

What an intercept means in practice

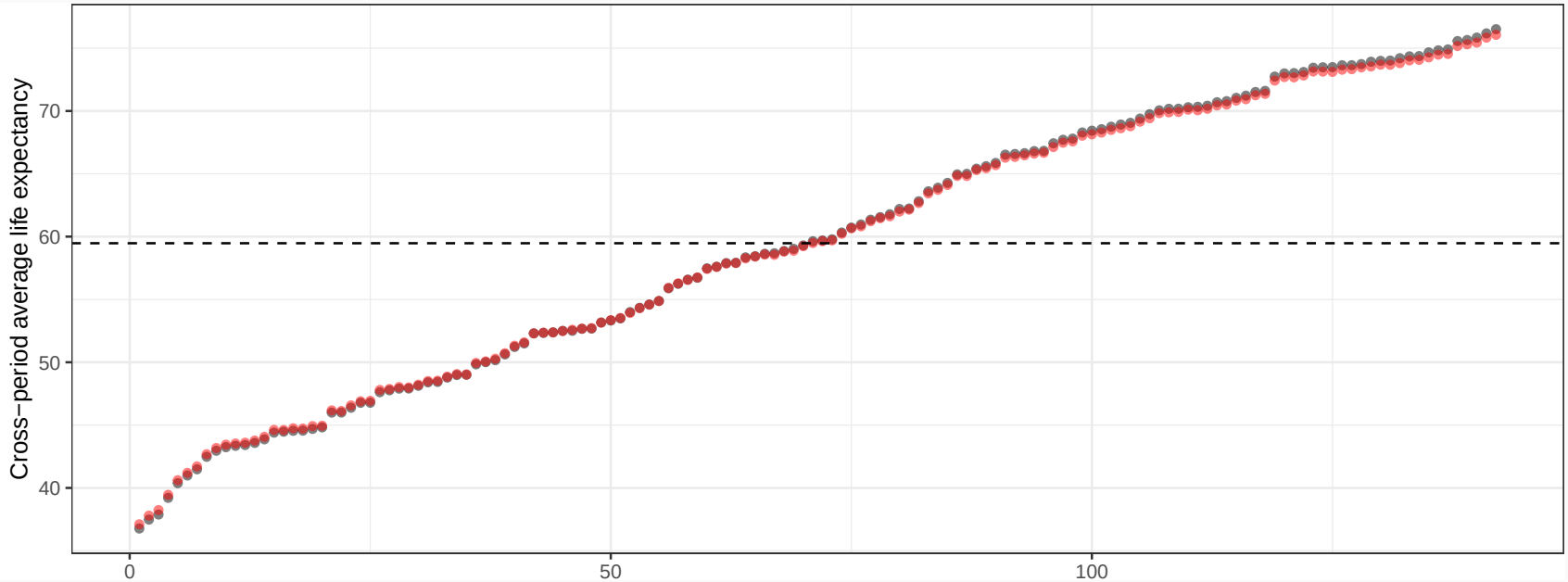
Intercepts are the weighted average of the *empirical mean* of the group and the prior (remember Bayes' theorem!). In this case, for country j

$$\alpha_j \propto \bar{L}_j \times p(\alpha_j)$$

```
sim_dat <- data.frame(country = unique(dat$country))
### draw posterior predictions of L for each country
sims <- link(m_LE1, sim_dat)
sims_PI<-apply(sims, 2, PI)
### compute empirical means, attach posterior estimates of \alpha[country]
plot_dat<-dat %>%
  group_by(country) %>%
  summarise(L_bar = mean(L)) %>%
  ungroup() %>%
  mutate(a_bar = apply(sims, 2, mean),
         a_lwr = sims_PI[1,],
         a_upr = sims_PI[2,]) %>%
  arrange(L_bar)
```

Bayesian regularization and empirical means

```
ggplot(plot_dat,
       aes(x = 1:nrow(plot_dat),
           y = L_bar)) +
  geom_point(alpha = 0.5) +
  geom_point(aes(y = a_bar), color = "red", alpha = 0.5) +
  geom_hline(yintercept = mean(plot_dat$L_bar), lty = 2) +
  labs(x = "", y = "Cross-period average life expectancy")
```



Regularization

Our model *regularizes* intercept estimates by gently nudging them toward the prior.

This shift is larger when:

- The prior is narrow (strong)
- The group mean is far away from the global mean
- We have few observations for a unit

This shift is smaller when:

- The prior is wide (weak)
- The group mean is close to the global mean
- We have many observations for a unit

No-pooling models

Our current approach estimates country intercepts with two pieces of information:

1. The observed (maximum 12) years of data on country-level life expectancy, assumed to follow a Normal likelihood
2. Our prior that country-level life expectancy follows a Normal(55, 15)

These estimates ignore any information on life expectancy from *other* countries.

What goes into a no-pooling model

- In frequentist models, estimated intercepts are equal to empirical group means
- Often called a ‘fixed effects’ model
- Bayesian approach is similar, but with regularization from the prior
- This approach typically *overfits* the data, and makes for poor prediction, unless we have a lot of data on each unit

The partial pooling alternative

- The complete pooling approach assumes all units are the same
- The no-pooling approach assumes all units are different
- The *partial pooling* approach assumes that all units are different, but come from the same larger population

Partial pooling: the basic approach

By adding a little complexity to our model, we can split the difference between the complete pooling and no pooling approach.

- We will still estimate an intercept for each country
- But we'd also like to leverage information *across* countries
- After all, each country is a subset of the broader population of countries
- We achieve this by adding an *adaptive prior*, estimated from the data, to our intercept

Tadpole survival

Let's see how this works with data on the survival of reed frog tadpoles. The experiment evaluated how population density and size affected the likelihood that tadpoles were eaten by predators

```
data(reedfrogs)
head(reedfrogs)
```

```
##   density pred  size surv propsurv
## 1      10   no   big    9      0.9
## 2      10   no   big   10      1.0
## 3      10   no   big    7      0.7
## 4      10   no   big   10      1.0
## 5      10   no small    9      0.9
## 6      10   no small    9      0.9
```

The complete pooling approach

Let's estimate survival probability *surv* for the population of tanks with the following model. This model has one intercept for all tanks.

$$surv_i \sim \text{Binomial}(n, p)$$

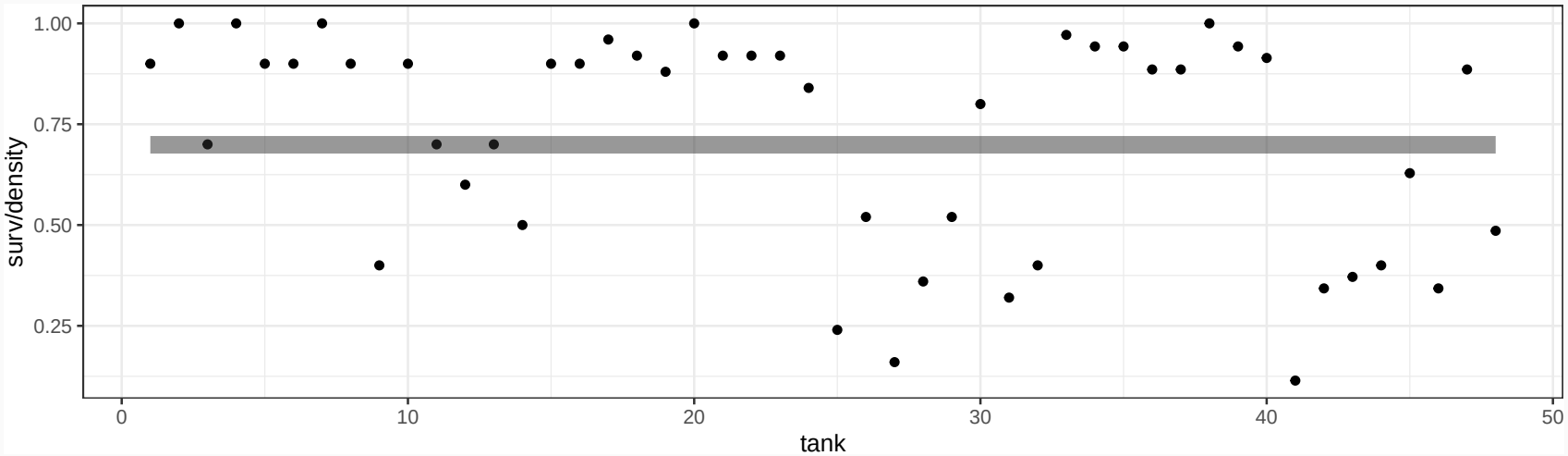
$$p = \alpha$$

$$\alpha \sim Normal(0, 1.5)$$

```
dat<-reedfrogs %>%
  mutate(tank = 1:nrow(reedfrogs)) %>%
  select(tank, surv, density)
m_tad_CP<-ulam(alist(
  surv ~ dbinom(density, p),
  logit(p) <- alpha,
  alpha ~ dnorm(0, 1.5)
), data = dat, chains = 4, log_lik = T)
```

Check the posterior for p

```
plot_dat<-dat
post_mu<-link(m_tad_CP, plot_dat)
plot_datCP<-plot_dat %>%
  mutate(mu_mn = apply(post_mu, 2, mean),
    mu_lwr = apply(post_mu, 2, PI)[1,],
         mu_upr = apply(post_mu, 2, PI)[2,])
ggplot(plot_datCP, aes(x = tank, y = surv/density)) +
  geom_point() +
  geom_ribbon(aes(ymin = mu_lwr, ymax = mu_upr), alpha = 0.5)
```



The complete pooling model

- Generates a very precise prediction
- Makes the same prediction for every tank in the pond
- *Underfits* the data. We can make much better predictions by estimating a more complex model.

The no pooling model

OK - so there are big differences among tanks, and the complete pooling approach underfits the data. Let's estimate tank-specific intercepts (fixed effects, dummy variable model)

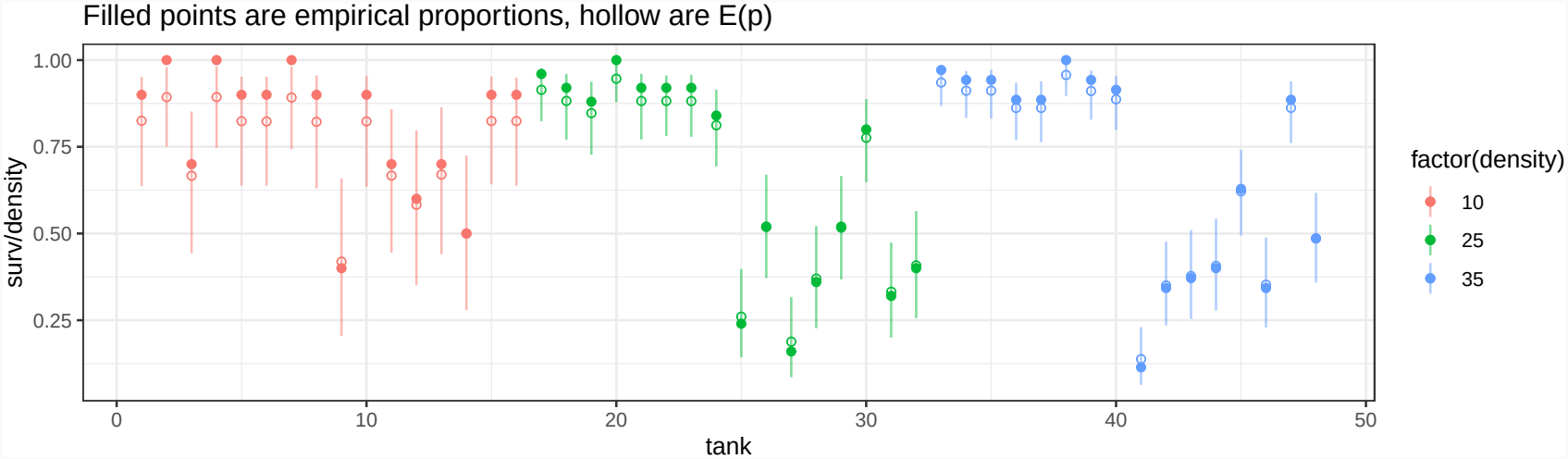
$$surv_i \sim \text{Binomial}(n, p)$$

$$p_i = \alpha[tank]$$

$$\alpha[tank] \sim Normal(0, 1.5)$$

```
m_tad_NP<-ulam(alist(
  surv ~ dbinom(density, p),
  logit(p) <- alpha[tank],
  alpha[tank] ~ dnorm(0, 1.5)
), data = dat, chains = 4, log_lik = T)
```

Check the posterior for *p*



The partial pooling (multilevel) approach

Now, we will use an adaptive prior for tank intercepts for each tank *j*. The prior for α has it's own prior, with *hyper-parameters* estimated from the data (!).

$$\text{surv}_i \sim \text{Binomial}(\text{dens}_i, p_i)$$

$$p_i = \alpha_j$$

$$\alpha_j \sim \text{Normal}(\bar{\alpha}, \sigma)$$

$$\bar{\alpha} \sim \text{Normal}(0, 1.5)$$

$$\sigma \sim \text{Exponential}(1)$$

Estimating the model

```
m_tad_PP<-ulam(alist(
  surv ~ dbinom(density, p),
  logit(p) <- a[tank],
  a[tank] ~ dnorm(a_bar, sigma),
  a_bar ~ dnorm(0, 1.5),
  sigma ~ dexp(1)),
  data = dat, chains = 4, cores = 4, log_lik = T)
```

Taking a look at what we've got here

```
precis(m_tad_PP)
```

```
##           mean        sd      5.5%    94.5%    n_eff     Rhat4
## a_bar 1.349055 0.2670679 0.9252207 1.789000 2780.259 0.9996396
## sigma 1.625447 0.2141874 1.3320280 1.991783 1311.806 1.0007516
```

Taking a look at what we've got here

```
precis(m_tad_PP)
```

```
##           mean        sd      5.5%    94.5%    n_eff     Rhat4
## a_bar 1.349055 0.2670679 0.9252207 1.789000 2780.259 0.9996396
## sigma 1.625447 0.2141874 1.3320280 1.991783 1311.806 1.0007516
```

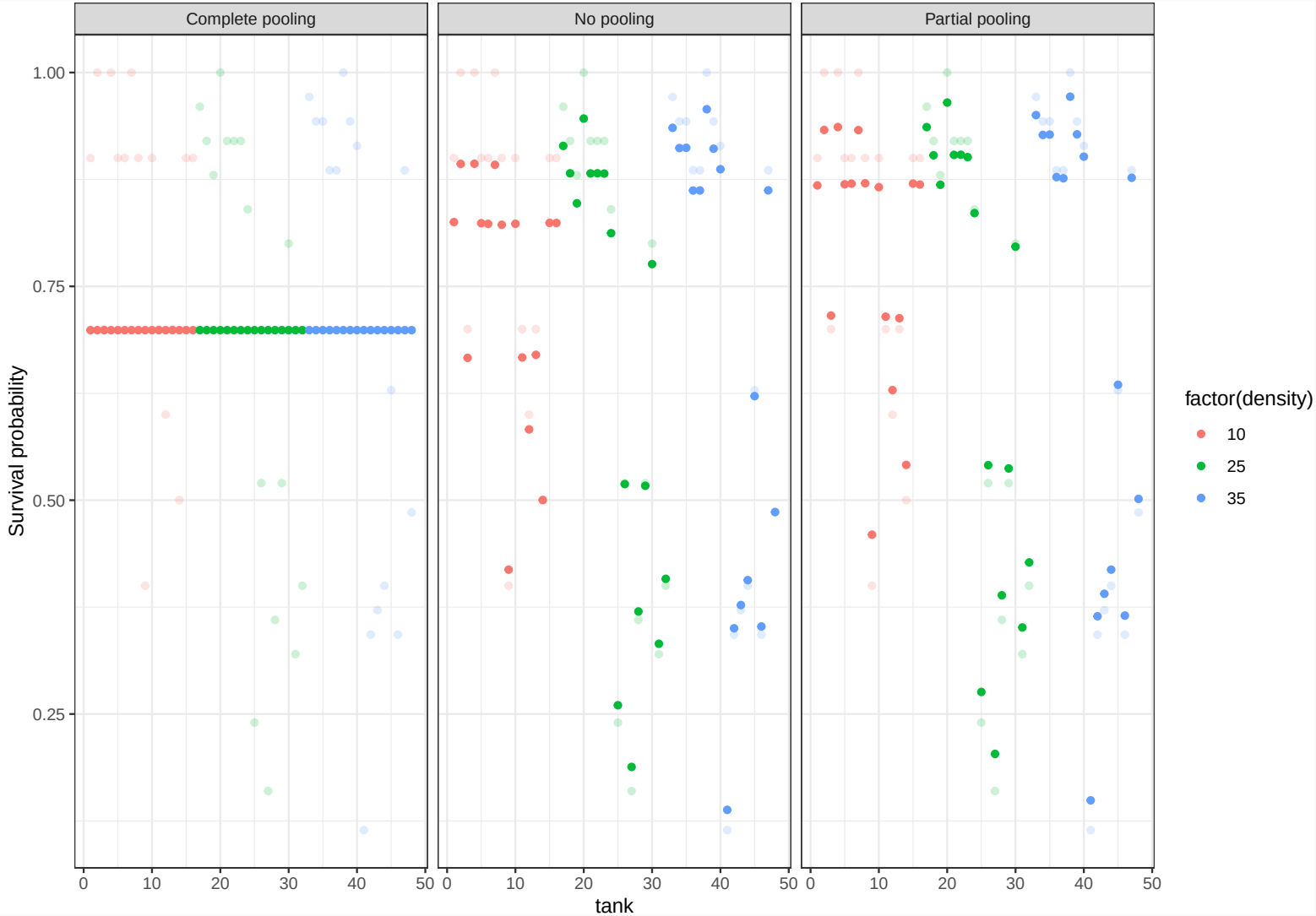
- $\bar{\alpha}$: the posterior for the center of the distribution of tank intercepts
- σ: the posterior for the standard deviation of tank intercepts

Taking a look at what we've got here: tank intercepts

```
precis(m_tad_PP, depth = 2)
```

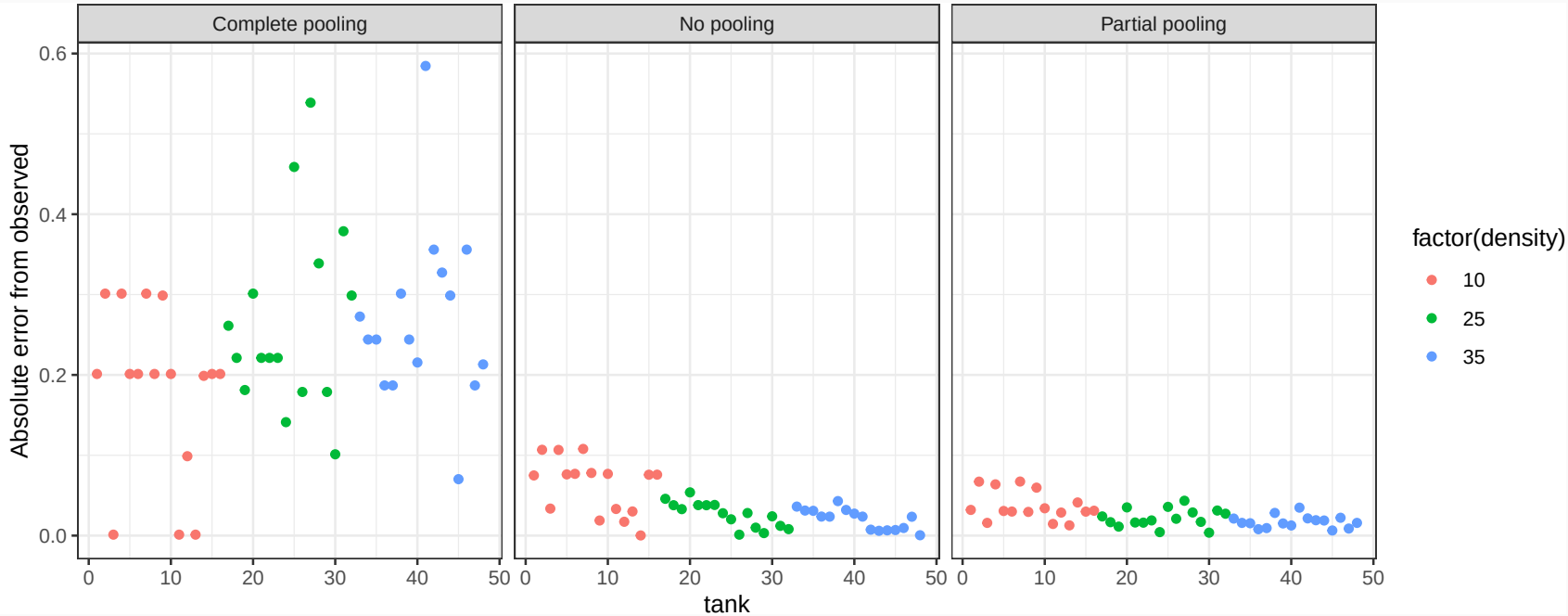
```
##               mean        sd         5.5%        94.5%    n_eff     Rhat4
## a[1]   2.128966173 0.8721354  0.849103824  3.567622540 2983.589 0.9990158
## a[2]   3.099389722 1.1480579  1.454929166  5.092222038 2745.846 1.0002518
## a[3]   1.015102791 0.6675354  0.006008356  2.115890951 3623.956 0.9991967
## a[4]   3.115899891 1.0957974  1.539536381  5.025584345 2772.244 1.0005733
## a[5]   2.136493833 0.8621080  0.889529866  3.588471913 3138.022 0.9985995
## a[6]   2.141328370 0.8534263  0.860253344  3.621331671 2737.405 0.9986278
## a[7]   3.124518173 1.1659684  1.415267091  5.090946884 2833.786 0.9986444
## a[8]   2.149740410 0.8620440  0.899879695  3.591065295 3340.745 0.9991674
## a[9]  -0.176847305 0.6055402 -1.125488627  0.780508662 3665.397 1.0007620
## a[10]  2.131855580 0.9097933  0.776712018  3.677366905 2455.824 0.9994101
## a[11]  1.002170506 0.6448555  0.009961720  2.077224405 4214.834 0.9996571
## a[12]  0.578714782 0.6566381 -0.470124826  1.659442205 4601.288 0.9983021
## a[13]  1.001443638 0.6812004 -0.048545240  2.143085699 4241.560 0.9992633
## a[14]  0.183465724 0.6699423 -0.853267697  1.244529412 3951.329 0.9988854
## a[15]  2.167931174 0.9080895  0.847616916  3.711799440 1861.113 0.9993617
## a[16]  2.130111116 0.8486989  0.882101627  3.590786338 2811.750 0.9988483
## a[17]  2.916880163 0.7839543  1.808750092  4.230871208 3288.240 0.9993913
## a[18]  2.408751015 0.6876623  1.413074995  3.546625696 2989.177 0.9988161
## a[19]  2.009150023 0.5858044  1.158182330  2.998664807 3396.660 1.0001177
## a[20]  3.691539710 1.0117422  2.351620711  5.423166356 1641.640 0.9998387
## a[21]  2.393752927 0.6440338  1.430027084  3.461383570 3944.337 0.9996218
## a[22]  2.410124099 0.6744036  1.417580733  3.580662728 3300.497 0.9989652
## a[23]  2.385996850 0.6932566  1.346147540  3.551353449 3558.185 0.9993386
## a[24]  1.712108706 0.5189945  0.927074749  2.576246666 4262.593 0.9995263
## a[25] -1.008561335 0.4459470 -1.738451245 -0.281931528 3667.014 0.9988666
## a[26]  0.170213233 0.3823300 -0.438026176  0.787005999 4522.232 0.9985217
```

Comparing model inferences



Comparing model inferences: error

```
ggplot(plot_dat_all,
       aes(x = tank,
           y = abs(surv/density - mu_mn),
           color = factor(density))) +
  geom_point() +
  facet_wrap(~model) +
  labs(y = "Absolute error from observed")
```



Comparing models: WAIC

```
compare(m_tad_CP, m_tad_NP, m_tad_PP)
```

```
##              WAIC        SE    dWAIC      dSE    pWAIC       weight
## m_tad_PP 200.7608  7.445671   0.0000       NA 21.35019 9.993659e-01
## m_tad_NP 215.4860  4.627279  14.7252  4.07801 26.02500 6.341428e-04
## m_tad_CP 584.1866 68.717103 383.4259 67.21949 10.39615 5.493612e-84
```

The impact of the partial pooling model on the posterior

- Ordinary Bayesian models *shrink* posterior estimates toward the prior
- Multilevel models shrink posterior estimates toward the group or cluster mean
- The effect is less pronounced when there is more data in a cluster
- When we have a lot of data, the partial pooling estimate is indistinguishable from the no pooling estimate

Clustering at more than one level

How many levels of clustering are there here?

```
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

Let's start with a model with varying country intercepts

We'll account for change over time with a simple linear term. We'll center and scale life expectancy as *L*. This makes priors easier, and reduces the number of parameters needed (because α will center near zero by definition).

$$L_i \sim Normal(\mu, \sigma)$$

$$\mu_i = \alpha_{[country]i} + \beta \times year_i$$

$$\alpha_{[country]} \sim \text{Normal}(0, \sigma_\alpha)$$

$$\beta \sim \text{Normal}(0, 2)$$

$$\sigma_\alpha \sim \text{Exponential}(1)$$

$$\sigma \sim \text{Exponential}(1)$$

Estimating the model

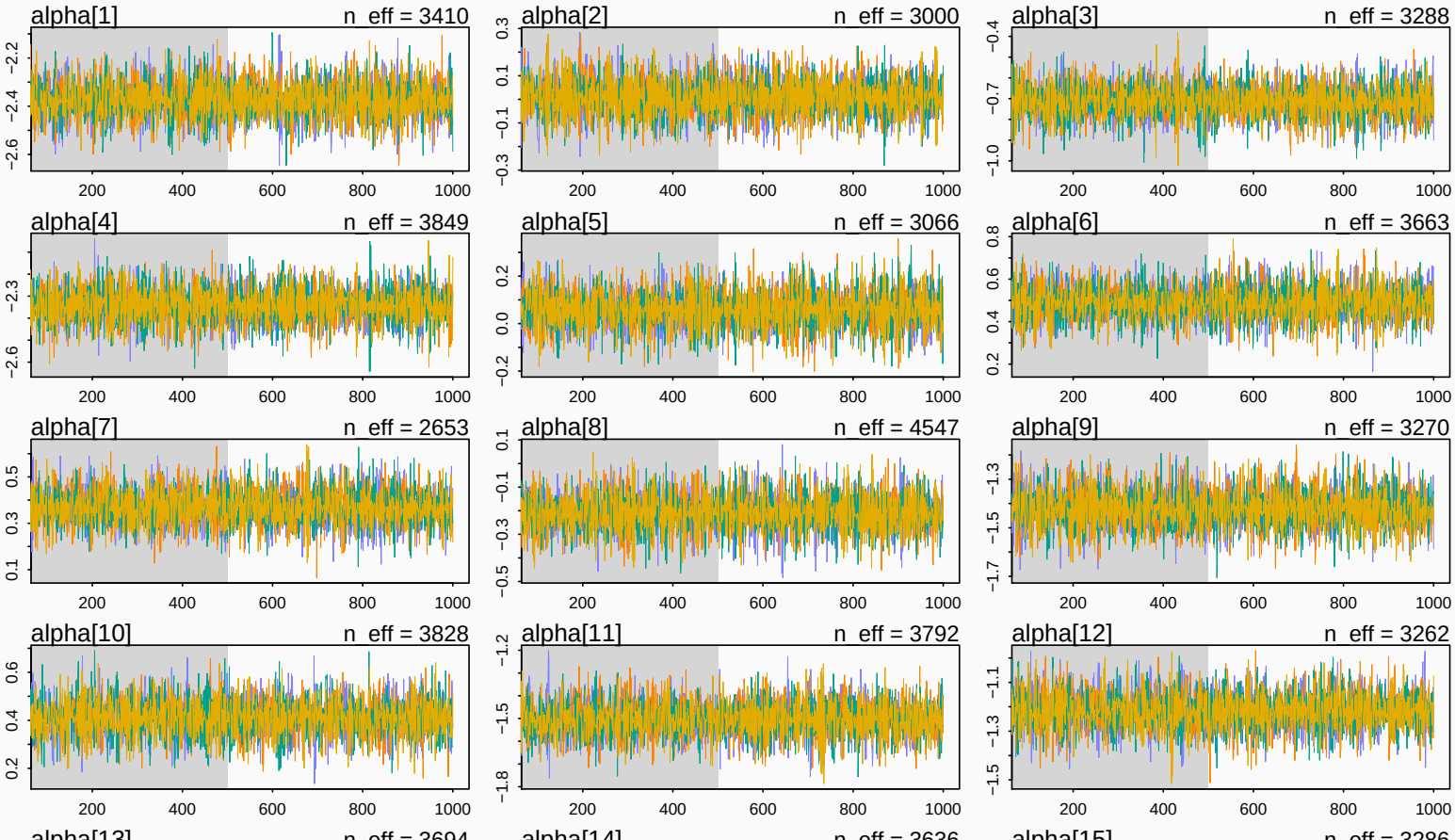
```
dat<-gapminder %>%
  mutate(L_c = (lifeExp - mean(lifeExp)) / sd(lifeExp),
         country = factor(country),
         year_c = (year - min(year))/5,
         continent = factor(continent)) %>%
  select(lifeExp, L_c, country, year, year_c, continent)

m_gm1<-ulam(alist(
  L_c ~ dnorm(mu, sigma),
  mu <- alpha[country] + beta * year_c,
  alpha[country] ~ dnorm(0, sigma_a),
  beta ~ dnorm(0, 2),
  sigma_a ~ dexp(1),
  sigma ~ dexp(1)
), data = dat, chains = 4, cores = 4, log_lik = T)
```

Checking the posterior

```
precis(m_gm1)
```

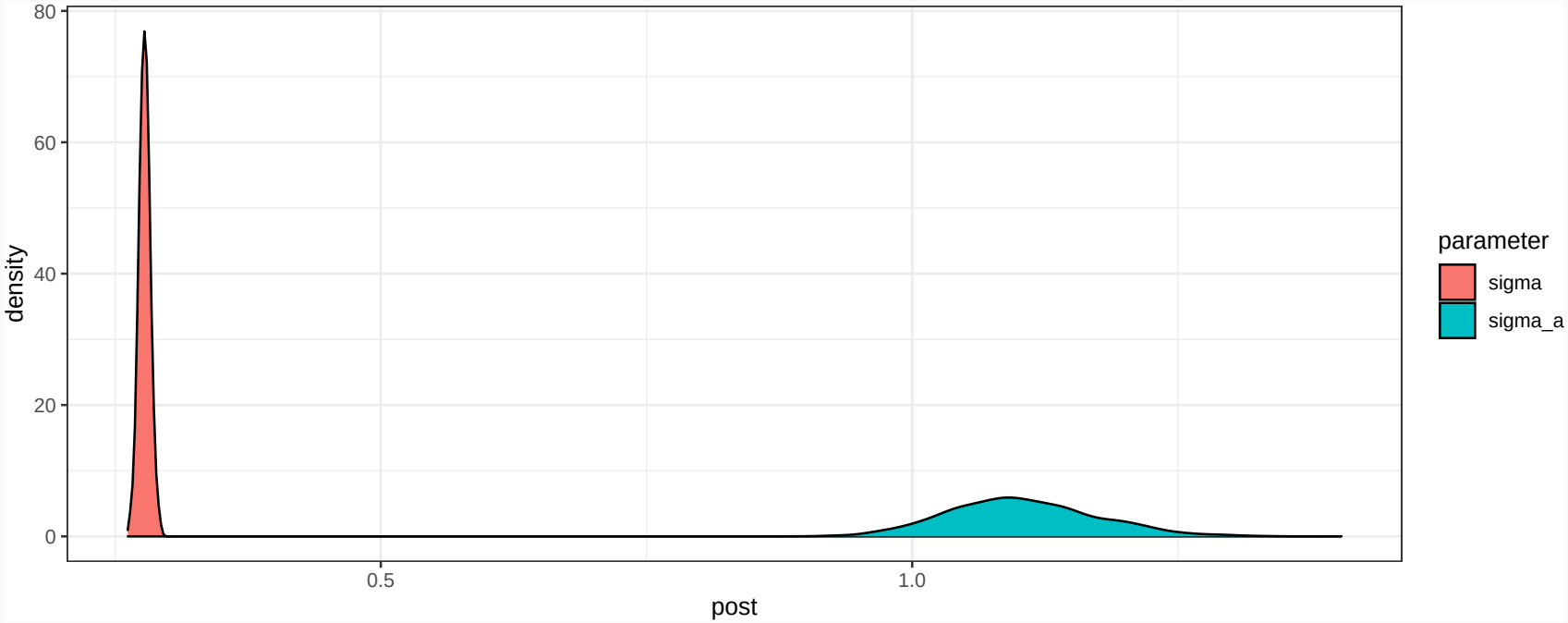
```
##               mean           sd      5.5%     94.5%    n_eff      Rhat4
## beta     0.1245831 0.002002710 0.1214515 0.1278420 1042.651 1.0006152
## sigma_a 1.1035904 0.068980895 0.9986397 1.2196001 4716.934 0.9992656
## sigma    0.2777380 0.005133597 0.2696370 0.2860621 3112.058 0.9997971
```

```
traceplot(m_gm1)
```



Within-cluster and between-cluster variance

```
post<-extract.samples(m_gm1)
post_plot<- data.frame(
  post = c(post$sigma, post$sigma_a),
  parameter = rep(c("sigma", "sigma_a"),
                  each = nrow(post[[1]])))

ggplot(post_plot, aes(x = post, fill = parameter)) +
  geom_density()
```



Adding more than one cluster variable

We know that countries are nested within continents. Perhaps much of the variation in country life expectancy is a function of continent-wide differences.

$$L_i \sim Normal(\mu, \sigma)$$

$$\mu_i = \alpha_{[country]i} + \gamma_{[continent]i} + \beta \times year_i$$

$$\alpha_{[country]} \sim \text{Normal}(0, \sigma_\alpha)$$

$$\gamma_{continent} \sim \text{Normal}(0, \sigma_\gamma$$

$$\beta \sim \text{Normal}(0, 2)$$

$$\sigma_\alpha \sim \text{Exponential}(1)$$

$$\sigma_\gamma \sim \text{Exponential}(1)$$

$$\sigma \sim \text{Exponential}(1)$$

s

Estimating the model

Note the increased iterations - this often helps when your Rhat or n_eff is low.

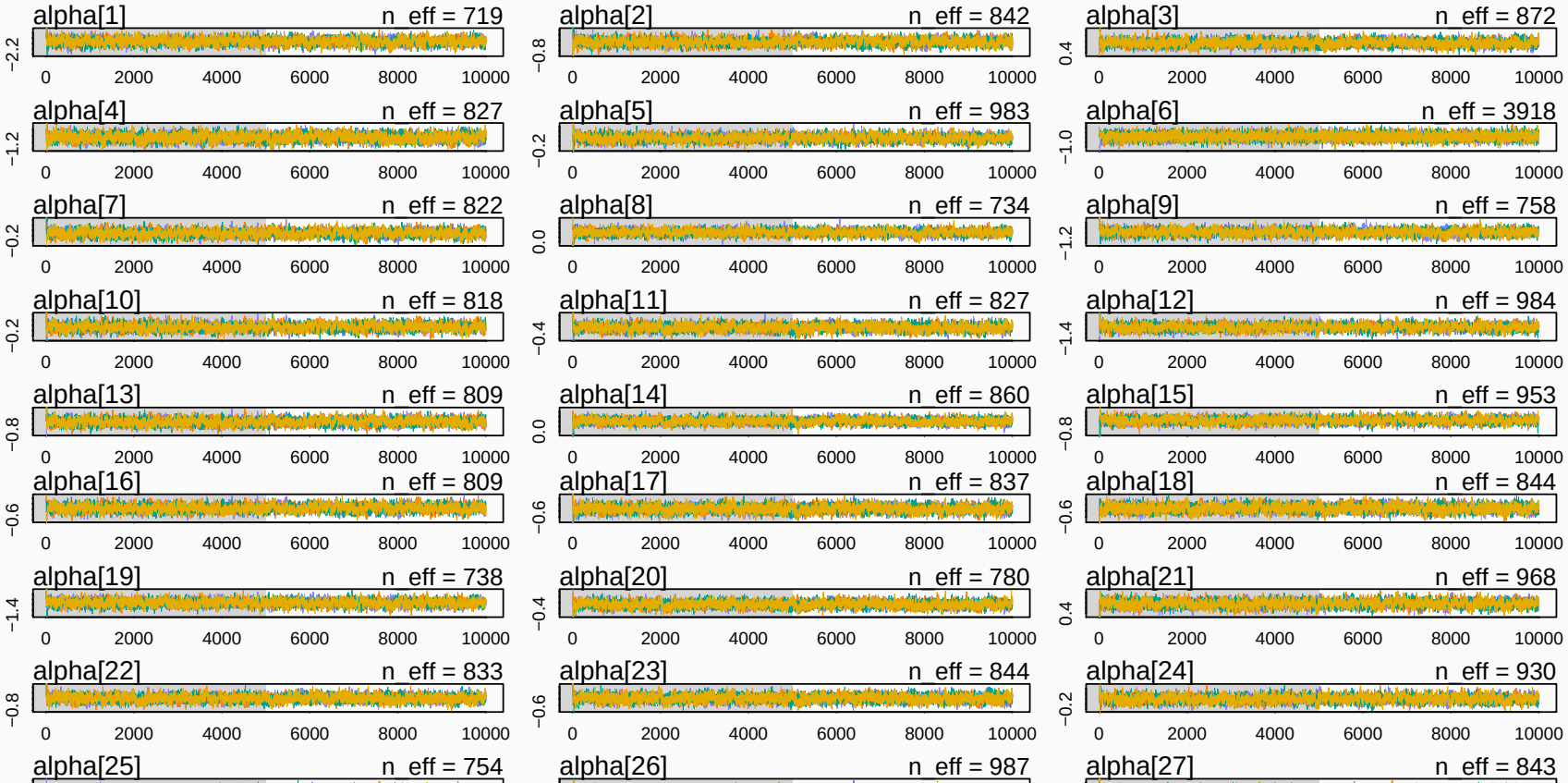
```
m_gm2<-ulam(alist(
  L_c ~ dnorm(mu, sigma),
  mu <- gamma[continent] + alpha[country] + beta * year_c,
  alpha[country] ~ dnorm(0, sigma_a),
  gamma[continent] ~ dnorm(0, sigma_g),
  beta ~ dnorm(0, 2),
  sigma_a ~ dexp(1),
  sigma_g ~ dexp(1),
  sigma ~ dexp(1)
), data = dat, chains = 4, cores = 4, iter = 10000, log_lik = T)
```

Checking the posterior and chains

```
precis(m_gm2)
```

```
##               mean           sd       5.5%      94.5%      n_eff     Rhat4
## beta    0.1260869 0.001954068 0.1229786 0.1292133 15413.653 1.000099
## sigma_a 0.5051469 0.031147084 0.4578479 0.5565135 10156.296 1.000217
## sigma_g 0.9175374 0.334267533 0.5392090 1.5236906  8214.552 1.000381
## sigma   0.2777026 0.004982001 0.2699151 0.2858857 11774.428 1.000438
```

```
traceplot(m_gm2)
```



Evaluating fit of the two models

```
compare(m_gm1, m_gm2)
```

```
##           WAIC       SE    dWAIC      dSE    pWAIC    weight
## m_gm2 615.8655 97.93811 0.000000       NA 137.1618 0.8078561
## m_gm1 618.7378 98.65644 2.872278 3.628568 139.2975 0.1921439
```

A bad model

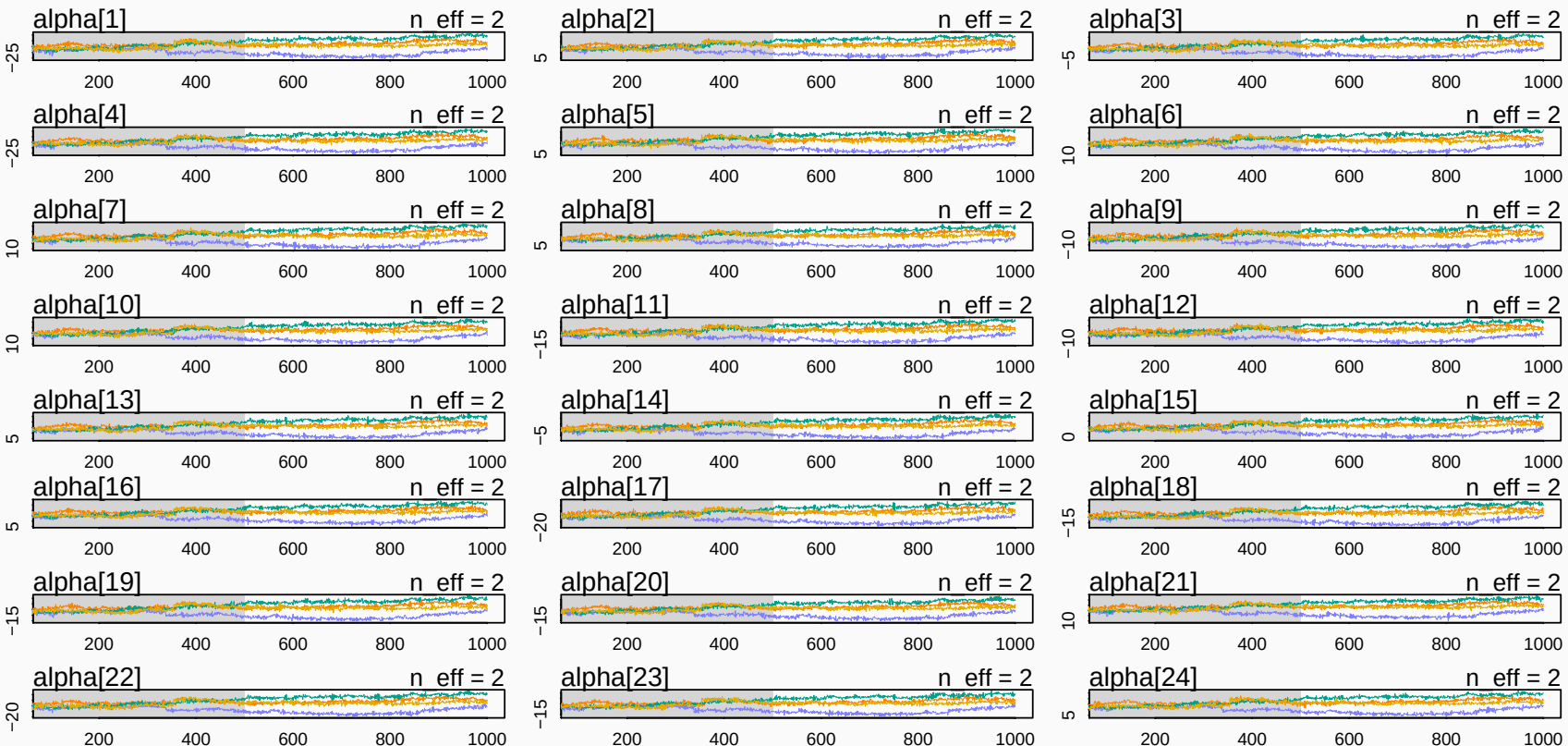
```
dat<-dat %>%
  mutate(year = factor(year))
m_bad<-ulam(alist(
  lifeExp ~ dnorm(mu, sigma),
  mu <- alpha[country] + gamma[year],
  alpha[country] ~ dnorm(alpha_bar, sigma_a),
  gamma[year] ~ dnorm(gamma_bar, sigma_g),
  alpha_bar ~ dnorm(55, 15),
  gamma_bar ~ dnorm(55, 15),
  sigma_a ~ dexp(0.2),
  sigma_g ~ dexp(0.2),
  sigma ~ dexp(0.2)
), data = dat, chains = 4, cores = 4)
```

Oof - this is what bad Markov chains look like

```
precis(m_bad)
```

```
##                  mean         sd       5.5%     94.5%       n_eff     Rhat4
## alpha_bar  8.221623 5.36673526 -1.494648 15.605477   2.248055 3.699664
## gamma_bar 51.469697 5.49978576 43.651713 61.346302   2.414233 2.696923
## sigma_a   11.082781 0.68259556 10.028493 12.208320 508.293855 1.006224
## sigma_g    6.230312 1.38943619  4.485854  8.667226 707.469854 1.001697
## sigma      3.457366 0.06134866  3.359040  3.552842 923.770845 1.000432
```

```
traceplot(m_bad)
```



Summary

Multilevel models are very flexible and useful for modeling:

- Repeat observations of units over time
- Data with clusters

Summary

Multilevel models are very flexible and useful for modeling:

- Repeat observations of units over time
- Data with clusters

Multilevel models

- Shrink estimates toward group means (like priors!)
- Provide variance estimates across and within clusters
- Allow for flexible posterior inference

Summary

Multilevel models are very flexible and useful for modeling:

- Repeat observations of units over time
- Data with clusters

Multilevel models

- Shrink estimates toward group means (like priors!)
- Provide variance estimates across and within clusters
- Allow for flexible posterior inference

Homework

- Chapter 13 easy questions, and 13H2