Git collaboration and hierarchical models

Harriet Ware

March 12 2021

```r
library(tidyverse)
library(rstan)
library(bayesplot)
```

1 Git collaboration

See github.

2 Radon

The goal of this lab is to fit this model to the radon data:

$$\begin{aligned} y_i|\alpha_{j[i]} &\sim N\left(\alpha_{j[i]} + \beta x_i, \sigma_y^2\right), \text{ for } i = 1, 2, \dots, n \\ \alpha_j &\sim N\left(\gamma_0 + \gamma_1 u_j, \sigma_\alpha^2\right), \text{ for } j = 1, 2, \dots, J \end{aligned}$$

i.e. varying intercepts, fixed slope on floor. I want you to

- reproduce the graph on slide 50.
- plot samples from the posterior predictive distribution for a new household in county 2 with basement level measurement, compared to samples from the posterior distribution of the mean county effect in county 2 (i.e., a graph similar to slide 39).

Here's code to get the data into a useful format:

```r
# house level data
d <- read.table(url("http://www.stat.columbia.edu/~gelman/arm/examples/radon/srrs2.dat"), header=T, sep

# deal with zeros, select what we want, make a fips variable to match on
d <- d %>%
  mutate(activity = ifelse(activity==0, 0.1, activity)) %>%
  mutate(fips = stfips * 1000 + cntyfips) %>%
  dplyr::select(fips, state, county, floor, activity)

# county level data
cty <- read.table(url("http://www.stat.columbia.edu/~gelman/arm/examples/radon/cty.dat"), header = T, s
cty <- cty %>% mutate(fips = 1000 * stfips + ctfips) %>% dplyr::select(fips, Uppm)
```

```r
# filter to just be minnesota, join them and then select the variables of interest.
dmn <- d %>%
  filter(state=="MN") %>%
  dplyr::select(fips, county, floor, activity) %>%
  left_join(cty)
head(dmn)
```

```
##    fips               county floor activity     Uppm
## 1 27001 AITKIN                   1      2.2 0.502054
## 2 27001 AITKIN                   0      2.2 0.502054
## 3 27001 AITKIN                   0      2.9 0.502054
## 4 27001 AITKIN                   0      1.0 0.502054
## 5 27003 ANOKA                    0      3.1 0.428565
## 6 27003 ANOKA                    0      2.5 0.428565
```

Note, in the model:

- y_i is log(activity)
- x_i is floor
- u_i is log(Uppm)

So to complete this task successfully you will need to show me / produce:

- stan code for the model
- a plot like slide 39
- a plot like slide 50

Suggested steps

1. write Stan model (note, you will need samples from post pred distribution, either do in Stan or later in R)
2. Get data in stan format
3. Run the model
4. For α plot, get median estimates of alpha's, and the 2.5th and 97.5th percentiles. Also get the median (mean fine, easier to pull from summary) of the gamma0 and gamma1. You can then use `geom_abline()` to plot mean regression line.
5. For the predicted y plot, you will need your posterior predictive samples for y's and then just use `geom_density()`

```r
#Get data in stan format
dmn$y <- log(dmn$activity)
dmn$x <- dmn$floor
dmn$u <- log(dmn$Uppm)
u <- dmn%>%select(u)%>%distinct()%>%pull()
dmn$county_code <- as.numeric(as.factor(dmn$county))
J <- max(dmn$county_code)

stan_data <- list(N = nrow(dmn),
                  J = J,
                  county = dmn$county_code,
                  y = dmn$y,
                  x = dmn$floor,
                  u = u)
```

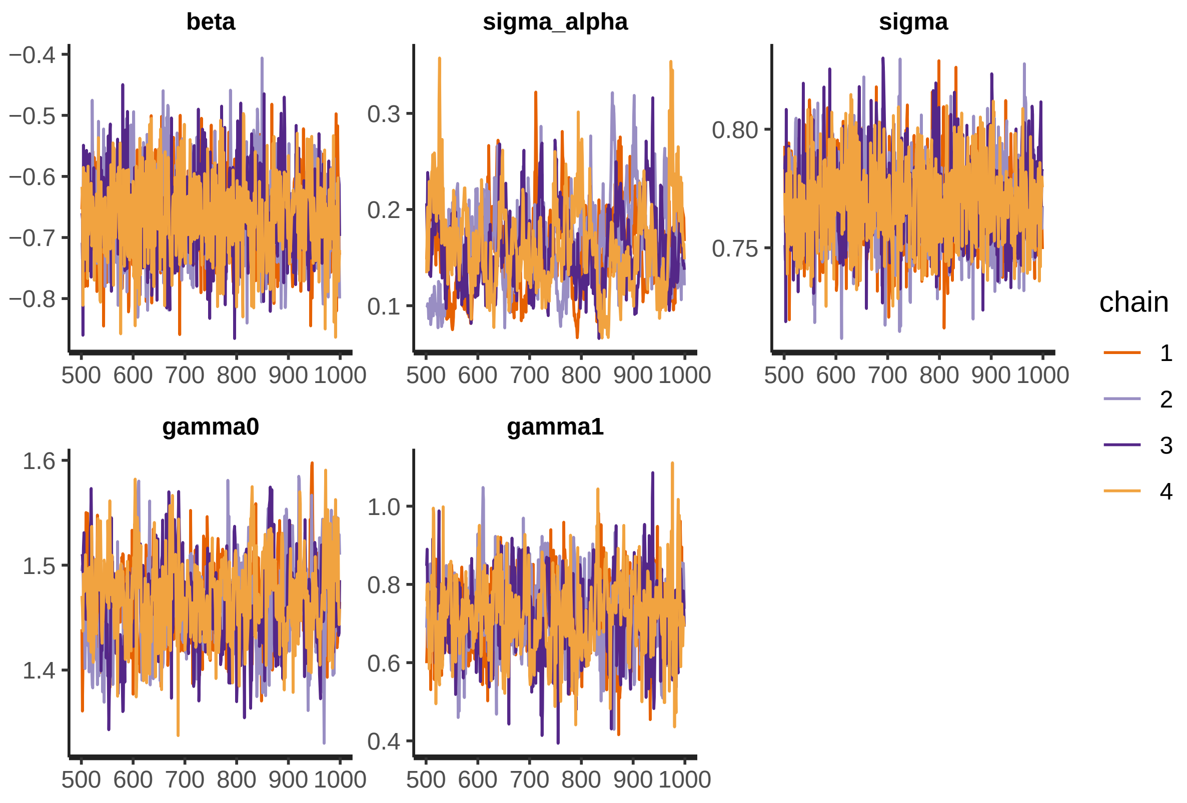
```r
#Run the model
mod <- stan(data = stan_data,
            file = "code/models/radon.stan",
            iter = 1000,
            seed = 243)
```

```
##
## SAMPLING FOR MODEL 'radon' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 4.239 seconds (Warm-up)
## Chain 1:                2.231 seconds (Sampling)
## Chain 1:                6.47 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'radon' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.277 seconds (Warm-up)
## Chain 2:                2.357 seconds (Sampling)
## Chain 2:                5.634 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'radon' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.463 seconds (Warm-up)
## Chain 3:                2.266 seconds (Sampling)
## Chain 3:                5.729 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'radon' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 3.473 seconds (Warm-up)
## Chain 4:                2.563 seconds (Sampling)
## Chain 4:                6.036 seconds (Total)
## Chain 4:
```

```r
summary(mod)$summary[c("beta", "sigma_alpha","sigma","gamma0","gamma1"),]
```

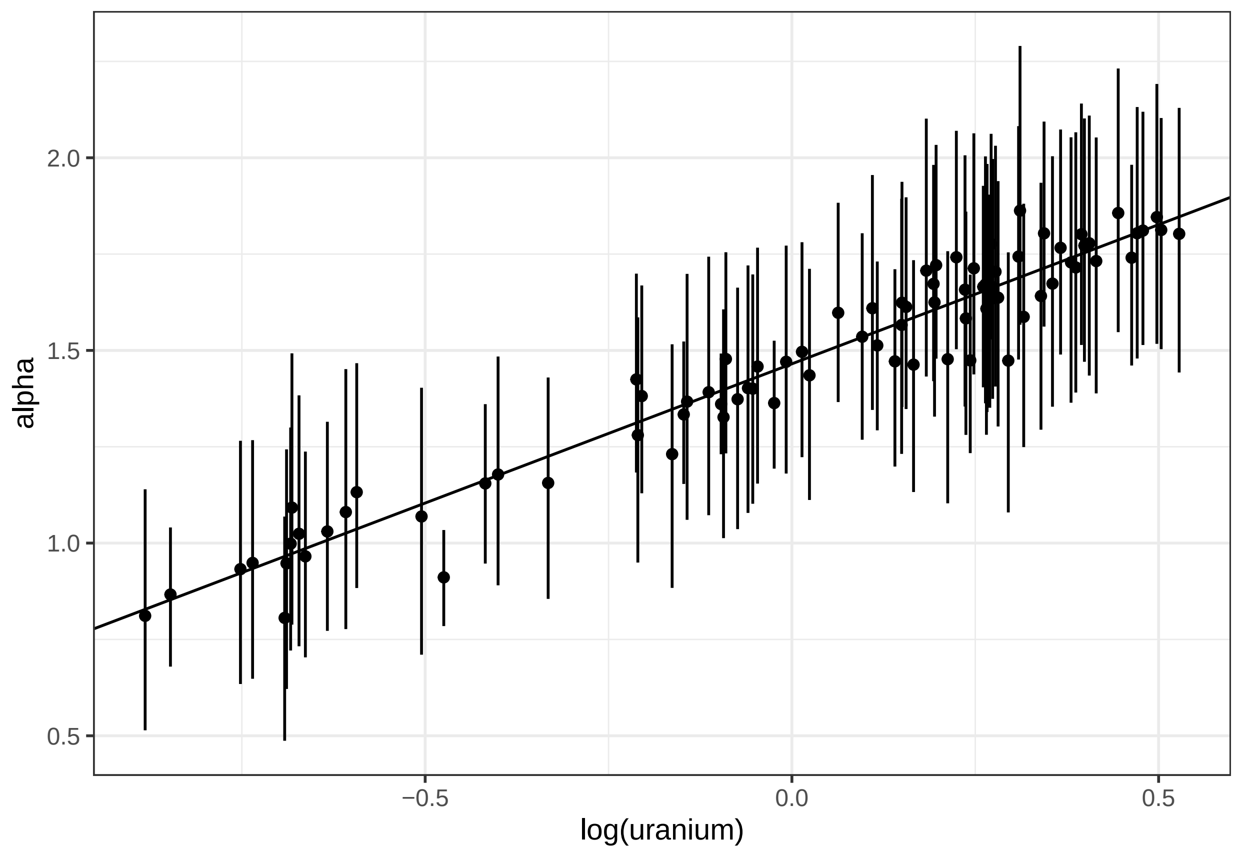
```
##                   mean      se_mean         sd        2.5%        25%
## beta        -0.6620501 0.0018896663 0.06832115 -0.79577330 -0.7089049
## sigma_alpha  0.1607886 0.0036673279 0.04487808  0.08687536  0.1286886
## sigma        0.7695669 0.0004884347 0.01839876  0.73592094  0.7573070
## gamma0       1.4649544 0.0016676569 0.03938839  1.38904937  1.4377590
## gamma1       0.7215914 0.0037590906 0.09538104  0.53278835  0.6606353
##                    50%        75%      97.5%     n_eff      Rhat
## beta        -0.6617192 -0.6175809 -0.5238640 1307.1941 1.0015385
## sigma_alpha  0.1569892  0.1889927  0.2605450  149.7507 1.0100029
## sigma        0.7688817  0.7814882  0.8067491 1418.9396 0.9995917
## gamma0       1.4651770  1.4902104  1.5453115  557.8573 1.0069482
## gamma1       0.7224890  0.7820377  0.9057047  643.8112 0.9989699
```

```r
traceplot(mod,pars=c("beta","sigma_alpha","sigma","gamma0","gamma1"))
```



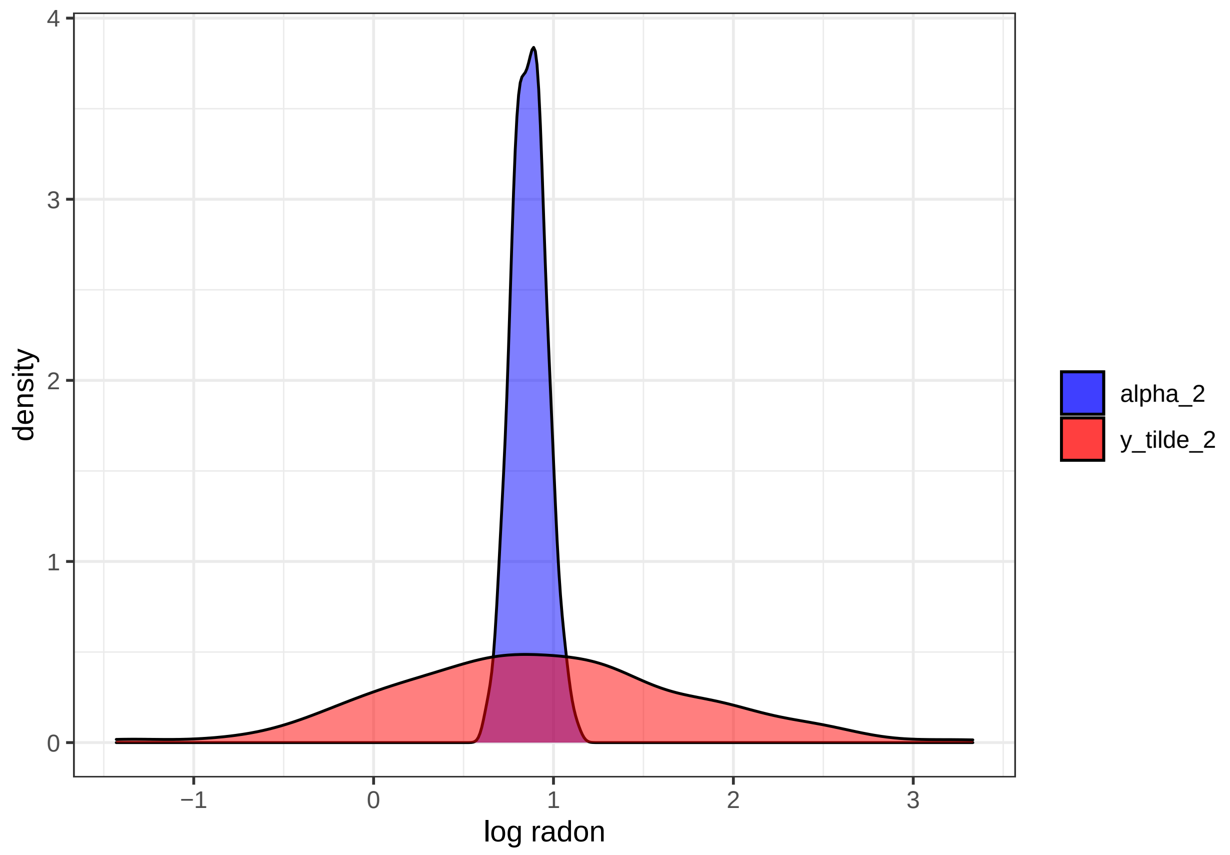
```r
#alpha plot like slide 50
alphas<-data.frame(summary(mod)$summary[6:90,c("50%","2.5%","97.5%")])
alphas$u=u
gamma0<-summary(mod)$summary[4,"50%"]
gamma1<-summary(mod)$summary[5,"50%"]

ggplot(alphas,aes(x=u,y=X50.))+geom_point()+
  geom_abline(intercept=gamma0,slope=gamma1)+
  geom_pointrange(aes(ymin=X2.5., ymax=X97.5.),fatten=1)+
  xlab("log(uranium)")+
  ylab("alpha")+
  theme_bw()
```



```
#predicted y plot like slide 39
post_samples <- extract(mod)
yrep <- post_samples$y_rep
sampy <- sample(yrep, 300)
alpha <- post_samples$alpha[,2]
sampa <- sample(alpha,300)

ggplot()+geom_density(aes(x=sampa,fill="blue"),alpha=0.5)+
  geom_density(aes(x=sampy,fill="red"),alpha=0.5)+
  xlab("log radon")+
  scale_fill_manual(name="",values = c("blue","red"),labels=c("alpha_2","y_tilde_2"))+
  theme_bw()
```


4
3
2
1
0
density
−1
0
1
2
3
log radon
alpha_2
y_tilde_2