Lab Exercise - Bayes

Luis Correia - Student No. 1006508566

February 26th 2020

Git collaboration

1. find a partner, add them as a collaborator to your class repo (you can/should remove them later once this is done)
2. create a text file in your repo with something in it
3. clone your partners repo, and **on a new branch** make changes to their text file
4. add, commit, push your changes on new branch upstream
5. do a pull request of your partner
6. accept your partners pull request

I'll be able to see the history.

Radon

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

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

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

- reproduce the graph on slide 43
- 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 32).

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

```
library(tidyverse)
library(rstan)
library(bayesplot) # PPCs
library(tidybayes) # may or may not be needed, but I like it
library(ggplot2)

# 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, makke 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, se
cty <- cty %>% mutate(fips = 1000 * stfips + ctfips) %>% dplyr::select(fips, Uppm)

# 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 sucessfully you will need to show me / produce:

- stan code for the model
- a plot like slide 32
- a plot like slide 43

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()`

Steps

2. Get data in stan format

```
# Generate a list of Countys in Minesotta
Ncty <- dmn %>%
  select(county, Uppm) %>%
  group_by(county, Uppm) %>%
  summarise()

# put into a list
stan_data <- list(N = nrow(dmn),
                  J = nrow(Ncty),
                  ctynb = match(dmn$county, Ncty$county),
```

```
                   x = dmn$floor,
                   y = log(dmn$activity),
                   u = log(Ncty$Uppm))

mod2 <- stan(data = stan_data,
             file = "STAN Model_v7.stan",
             iter = 250,
             seed = 530)
```

```
##
## SAMPLING FOR MODEL 'STAN Model_v7' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: There aren't enough warmup iterations to fit the
## Chain 1:          three stages of adaptation as currently configured.
## Chain 1:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 1:          the given number of warmup iterations:
## Chain 1:            init_buffer = 18
## Chain 1:            adapt_window = 95
## Chain 1:            term_buffer = 12
## Chain 1:
## Chain 1: Iteration:   1 / 250 [  0%]  (Warmup)
## Chain 1: Iteration:  25 / 250 [ 10%]  (Warmup)
## Chain 1: Iteration:  50 / 250 [ 20%]  (Warmup)
## Chain 1: Iteration:  75 / 250 [ 30%]  (Warmup)
## Chain 1: Iteration: 100 / 250 [ 40%]  (Warmup)
## Chain 1: Iteration: 125 / 250 [ 50%]  (Warmup)
## Chain 1: Iteration: 126 / 250 [ 50%]  (Sampling)
## Chain 1: Iteration: 150 / 250 [ 60%]  (Sampling)
## Chain 1: Iteration: 175 / 250 [ 70%]  (Sampling)
## Chain 1: Iteration: 200 / 250 [ 80%]  (Sampling)
## Chain 1: Iteration: 225 / 250 [ 90%]  (Sampling)
## Chain 1: Iteration: 250 / 250 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.574 seconds (Warm-up)
## Chain 1:                0.487 seconds (Sampling)
## Chain 1:                1.061 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'STAN Model_v7' 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: WARNING: There aren't enough warmup iterations to fit the
## Chain 2:          three stages of adaptation as currently configured.
## Chain 2:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 2:          the given number of warmup iterations:
```

```
## Chain 2:           init_buffer = 18
## Chain 2:           adapt_window = 95
## Chain 2:           term_buffer = 12
## Chain 2:
## Chain 2: Iteration:   1 / 250 [  0%]  (Warmup)
## Chain 2: Iteration:  25 / 250 [ 10%]  (Warmup)
## Chain 2: Iteration:  50 / 250 [ 20%]  (Warmup)
## Chain 2: Iteration:  75 / 250 [ 30%]  (Warmup)
## Chain 2: Iteration: 100 / 250 [ 40%]  (Warmup)
## Chain 2: Iteration: 125 / 250 [ 50%]  (Warmup)
## Chain 2: Iteration: 126 / 250 [ 50%]  (Sampling)
## Chain 2: Iteration: 150 / 250 [ 60%]  (Sampling)
## Chain 2: Iteration: 175 / 250 [ 70%]  (Sampling)
## Chain 2: Iteration: 200 / 250 [ 80%]  (Sampling)
## Chain 2: Iteration: 225 / 250 [ 90%]  (Sampling)
## Chain 2: Iteration: 250 / 250 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.633 seconds (Warm-up)
## Chain 2:                0.462 seconds (Sampling)
## Chain 2:                1.095 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'STAN Model_v7' 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: WARNING: There aren't enough warmup iterations to fit the
## Chain 3:          three stages of adaptation as currently configured.
## Chain 3:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 3:          the given number of warmup iterations:
## Chain 3:           init_buffer = 18
## Chain 3:           adapt_window = 95
## Chain 3:           term_buffer = 12
## Chain 3:
## Chain 3: Iteration:   1 / 250 [  0%]  (Warmup)
## Chain 3: Iteration:  25 / 250 [ 10%]  (Warmup)
## Chain 3: Iteration:  50 / 250 [ 20%]  (Warmup)
## Chain 3: Iteration:  75 / 250 [ 30%]  (Warmup)
## Chain 3: Iteration: 100 / 250 [ 40%]  (Warmup)
## Chain 3: Iteration: 125 / 250 [ 50%]  (Warmup)
## Chain 3: Iteration: 126 / 250 [ 50%]  (Sampling)
## Chain 3: Iteration: 150 / 250 [ 60%]  (Sampling)
## Chain 3: Iteration: 175 / 250 [ 70%]  (Sampling)
## Chain 3: Iteration: 200 / 250 [ 80%]  (Sampling)
## Chain 3: Iteration: 225 / 250 [ 90%]  (Sampling)
## Chain 3: Iteration: 250 / 250 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.622 seconds (Warm-up)
## Chain 3:                0.533 seconds (Sampling)
## Chain 3:                1.155 seconds (Total)
## Chain 3:
```

```
##
## SAMPLING FOR MODEL 'STAN Model_v7' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.001 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: WARNING: There aren't enough warmup iterations to fit the
## Chain 4:          three stages of adaptation as currently configured.
## Chain 4:          Reducing each adaptation stage to 15%/75%/10% of
## Chain 4:          the given number of warmup iterations:
## Chain 4:            init_buffer = 18
## Chain 4:            adapt_window = 95
## Chain 4:            term_buffer = 12
## Chain 4:
## Chain 4: Iteration:   1 / 250 [  0%]  (Warmup)
## Chain 4: Iteration:  25 / 250 [ 10%]  (Warmup)
## Chain 4: Iteration:  50 / 250 [ 20%]  (Warmup)
## Chain 4: Iteration:  75 / 250 [ 30%]  (Warmup)
## Chain 4: Iteration: 100 / 250 [ 40%]  (Warmup)
## Chain 4: Iteration: 125 / 250 [ 50%]  (Warmup)
## Chain 4: Iteration: 126 / 250 [ 50%]  (Sampling)
## Chain 4: Iteration: 150 / 250 [ 60%]  (Sampling)
## Chain 4: Iteration: 175 / 250 [ 70%]  (Sampling)
## Chain 4: Iteration: 200 / 250 [ 80%]  (Sampling)
## Chain 4: Iteration: 225 / 250 [ 90%]  (Sampling)
## Chain 4: Iteration: 250 / 250 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.543 seconds (Warm-up)
## Chain 4:                0.482 seconds (Sampling)
## Chain 4:                1.025 seconds (Total)
## Chain 4:
```

```
summary(mod2)$summary[c(paste0("alpha[", 1:85, "]"), "gamma0", "gamma1"),]
```

```
##                  mean     se_mean         sd      2.5%       25%       50%
## alpha[1]  0.9437248 0.006791997 0.16347404 0.6003086 0.8377980 0.9469779
## alpha[2]  0.8641150 0.003234014 0.09221455 0.6960547 0.8086579 0.8646156
## alpha[3]  1.4064042 0.007414445 0.17356441 1.0807829 1.2937429 1.3930880
## alpha[4]  1.1605203 0.010559785 0.16731556 0.8501004 1.0547476 1.1471968
## alpha[5]  1.3791927 0.007004241 0.16292684 1.0621965 1.2645434 1.3884649
## alpha[6]  1.7163197 0.009746791 0.18364821 1.2895640 1.6033730 1.7256593
## alpha[7]  1.8005709 0.008072429 0.13588821 1.5314638 1.7091662 1.7963990
## alpha[8]  1.7327580 0.008201461 0.17554360 1.4048523 1.6248214 1.7286016
## alpha[9]  1.1497597 0.005827155 0.15238413 0.8445415 1.0572807 1.1461695
## alpha[10] 1.5332565 0.004683737 0.13996192 1.2489477 1.4449929 1.5375115
## alpha[11] 1.1020384 0.009451612 0.16188240 0.7882951 1.0037712 1.0944192
## alpha[12] 1.6810678 0.008474758 0.17930363 1.3334187 1.5713881 1.6805269
## alpha[13] 0.9653882 0.007645077 0.16382807 0.6619180 0.8593702 0.9604603
## alpha[14] 1.8162024 0.006614593 0.13468225 1.5742209 1.7200427 1.8159561
## alpha[15] 1.4069012 0.006776763 0.16255736 1.0873947 1.2935416 1.4142436
## alpha[16] 1.0583781 0.008951924 0.17433827 0.7201759 0.9586178 1.0657136
## alpha[17] 1.6225920 0.010924473 0.15608471 1.3000704 1.5241649 1.6320559
## alpha[18] 1.0514140 0.006287497 0.15023461 0.7926689 0.9512169 1.0420022
```

```
## alpha[19] 1.3598405 0.003140339 0.08494697 1.1886015 1.3013978 1.3580896
## alpha[20] 1.6795396 0.007838278 0.18138610 1.3376513 1.5536021 1.6688453
## alpha[21] 1.6293765 0.005980928 0.14987388 1.3735246 1.5229271 1.6212164
## alpha[22] 1.4369350 0.021305445 0.19644142 0.9777059 1.3306161 1.4578526
## alpha[23] 1.7390610 0.008175252 0.17274986 1.3688676 1.6377156 1.7444353
## alpha[24] 1.7787542 0.009230607 0.15348277 1.4861867 1.6774506 1.7735727
## alpha[25] 1.7470737 0.008315168 0.14426807 1.4801567 1.6444507 1.7389605
## alpha[26] 1.3617622 0.002790698 0.06425662 1.2398157 1.3198699 1.3605853
## alpha[27] 1.8210058 0.006557154 0.17193020 1.4865075 1.7155880 1.8172945
## alpha[28] 1.1748966 0.006702670 0.16482005 0.8669799 1.0685205 1.1694541
## alpha[29] 0.9422528 0.008326841 0.20154376 0.5783646 0.8114350 0.9463985
## alpha[30] 0.9657319 0.005792974 0.16005316 0.6591570 0.8638792 0.9636989
## alpha[31] 1.7604714 0.007941514 0.16847847 1.4400949 1.6477122 1.7614930
## alpha[32] 1.3903951 0.006058198 0.14961872 1.1001395 1.2927981 1.3893751
## alpha[33] 1.6403317 0.010239541 0.17710373 1.2926143 1.5289762 1.6394741
## alpha[34] 1.4845830 0.006291087 0.16984230 1.1531679 1.3767220 1.4687776
## alpha[35] 0.8111007 0.007224788 0.14596338 0.5280406 0.7093017 0.8170065
## alpha[36] 1.9048039 0.015598808 0.18991013 1.5529179 1.7815380 1.8823189
## alpha[37] 0.7847717 0.011265672 0.16882158 0.4385894 0.6792396 0.7908979
## alpha[38] 1.1152515 0.015039365 0.17646640 0.8168806 0.9959699 1.0983287
## alpha[39] 1.6350967 0.006672588 0.16930096 1.2917579 1.5260263 1.6329312
## alpha[40] 1.8683686 0.009158854 0.17745115 1.5695705 1.7355277 1.8610985
## alpha[41] 1.8154838 0.005859756 0.15575735 1.5247686 1.7073649 1.8055379
## alpha[42] 1.5675556 0.007625627 0.17334511 1.2473756 1.4607899 1.5660465
## alpha[43] 1.5074343 0.006096628 0.15123922 1.2167331 1.4052859 1.5041576
## alpha[44] 1.4413065 0.010104247 0.16145938 1.1179447 1.3377530 1.4504868
## alpha[45] 1.4592093 0.007543030 0.14693251 1.1364930 1.3714466 1.4615720
## alpha[46] 1.4377164 0.007000727 0.15375524 1.1032505 1.3339161 1.4597017
## alpha[47] 1.2747186 0.008117174 0.17535139 0.9053672 1.1696042 1.2825267
## alpha[48] 1.3178787 0.005473260 0.14391164 1.0218632 1.2324390 1.3229702
## alpha[49] 1.6703804 0.005182746 0.13662925 1.4066351 1.5804360 1.6750360
## alpha[50] 1.8024001 0.009407977 0.18404465 1.4399833 1.6841035 1.8017805
## alpha[51] 1.7408135 0.010129885 0.17870782 1.4327886 1.6099583 1.7301346
## alpha[52] 1.7902816 0.009186072 0.17049818 1.4430271 1.6837842 1.7963543
## alpha[53] 1.5953374 0.007151487 0.17844449 1.1750752 1.4942171 1.6095596
## alpha[54] 1.4639921 0.007426050 0.12323720 1.1989033 1.3886733 1.4678726
## alpha[55] 1.4038736 0.008439110 0.15549907 1.1341121 1.2949745 1.4010441
## alpha[56] 1.3647668 0.006921676 0.17132234 1.0133300 1.2659854 1.3604093
## alpha[57] 1.2021681 0.011536383 0.15984331 0.8546996 1.1053151 1.2248118
## alpha[58] 1.8353645 0.008358129 0.17688702 1.5015822 1.7227362 1.8238880
## alpha[59] 1.6666605 0.007629792 0.17367686 1.2964955 1.5589713 1.6622758
## alpha[60] 1.6374820 0.008879562 0.19228269 1.2594266 1.5148993 1.6341712
## alpha[61] 1.1555056 0.002943556 0.10536955 0.9483637 1.0902712 1.1608764
## alpha[62] 1.7825988 0.007759836 0.16415492 1.4760881 1.6736342 1.7801023
## alpha[63] 1.7370151 0.009968460 0.18454479 1.3459026 1.6115102 1.7381488
## alpha[64] 1.6942756 0.004913000 0.14986331 1.4188702 1.5981748 1.6930269
## alpha[65] 1.7962619 0.011031180 0.18362459 1.4096724 1.6759956 1.7989684
## alpha[66] 1.4454354 0.007502647 0.13942484 1.1835332 1.3419194 1.4389416
## alpha[67] 1.6116030 0.006381608 0.14064832 1.3579479 1.5182761 1.6014727
## alpha[68] 1.0045706 0.005821297 0.15124371 0.7300371 0.9045835 0.9930111
## alpha[69] 1.5731124 0.008276833 0.16770447 1.2588701 1.4555666 1.5742713
## alpha[70] 0.9019086 0.003198241 0.06863647 0.7718581 0.8590990 0.9034781
## alpha[71] 1.5123063 0.003980587 0.11288527 1.2986647 1.4289370 1.5166970
## alpha[72] 1.6482704 0.004437031 0.14264302 1.3438983 1.5536165 1.6497846
```

```
## alpha[73] 1.8202625 0.008014078 0.16546987 1.5069371 1.7141595 1.8103307
## alpha[74] 1.5789974 0.013786371 0.17210378 1.2135882 1.4714678 1.5934878
## alpha[75] 1.4714608 0.006891629 0.16415286 1.1376240 1.3663524 1.4659534
## alpha[76] 1.8565096 0.007535286 0.16445985 1.5422015 1.7483991 1.8500788
## alpha[77] 1.6414789 0.005885955 0.14935315 1.3511598 1.5598113 1.6360817
## alpha[78] 1.0264706 0.008466681 0.17450429 0.7010580 0.9068194 1.0288979
## alpha[79] 1.4407644 0.013454064 0.18722739 1.0379354 1.3357802 1.4541670
## alpha[80] 1.3279307 0.003346932 0.09608027 1.1533183 1.2574433 1.3288943
## alpha[81] 1.7487296 0.010002195 0.18276099 1.4065582 1.6327622 1.7389696
## alpha[82] 1.6842955 0.010109824 0.17933497 1.3548943 1.5687378 1.6797961
## alpha[83] 1.7268814 0.006359192 0.15579606 1.4128213 1.6207355 1.7297022
## alpha[84] 1.4915983 0.005201275 0.13013416 1.2434164 1.4038964 1.4822317
## alpha[85] 1.6693657 0.011380861 0.18545610 1.3063659 1.5507704 1.6847219
## gamma0     1.4693273 0.002956885 0.04101152 1.3913669 1.4423267 1.4677190
## gamma1     0.7311968 0.007691869 0.09777246 0.5387533 0.6611365 0.7343000
##                  75%    97.5%      n_eff      Rhat
## alpha[1]   1.0432042 1.268855  579.29874 0.9956687
## alpha[2]   0.9186631 1.048378  813.04617 0.9991657
## alpha[3]   1.5125812 1.797511  547.97925 0.9970358
## alpha[4]   1.2622126 1.528014  251.05130 1.0160407
## alpha[5]   1.4771278 1.699586  541.08209 0.9938719
## alpha[6]   1.8316144 2.052649  355.01776 1.0085639
## alpha[7]   1.8882192 2.090689  283.37079 1.0060907
## alpha[8]   1.8232772 2.108264  458.12870 1.0006078
## alpha[9]   1.2542180 1.445985  683.85867 1.0013015
## alpha[10] 1.6255016 1.804978  892.96586 0.9981458
## alpha[11] 1.2033624 1.424208  293.35095 1.0022456
## alpha[12] 1.7881997 2.057890  447.63448 0.9979573
## alpha[13] 1.0649341 1.295861  459.21162 1.0010504
## alpha[14] 1.9035452 2.078118  414.58580 1.0043603
## alpha[15] 1.5203667 1.721194  575.39842 1.0011050
## alpha[16] 1.1705782 1.375904  379.27365 1.0109606
## alpha[17] 1.7234421 1.907576  204.13606 1.0286409
## alpha[18] 1.1307274 1.389895  570.93195 1.0014002
## alpha[19] 1.4180245 1.536973  731.71624 0.9973303
## alpha[20] 1.7967506 2.039578  535.50886 1.0006681
## alpha[21] 1.7398666 1.930657  627.93501 0.9962793
## alpha[22] 1.5685132 1.762768   85.01292 1.0529634
## alpha[23] 1.8613530 2.074374  446.51193 0.9995117
## alpha[24] 1.8694428 2.079746  276.47682 1.0070983
## alpha[25] 1.8402847 2.054914  301.02212 1.0132578
## alpha[26] 1.4043682 1.487886  530.16379 1.0096813
## alpha[27] 1.9378454 2.161613  687.50137 0.9996795
## alpha[28] 1.2881121 1.503661  604.67834 0.9956692
## alpha[29] 1.0644385 1.311187  585.83885 0.9970451
## alpha[30] 1.0714307 1.286953  763.35277 0.9947232
## alpha[31] 1.8670187 2.097768  450.07218 1.0031526
## alpha[32] 1.4894601 1.678001  609.93689 1.0018184
## alpha[33] 1.7567504 1.989973  299.15373 1.0029853
## alpha[34] 1.5837258 1.836969  728.85355 0.9971491
## alpha[35] 0.9031305 1.104959  408.16669 1.0050182
## alpha[36] 2.0166653 2.309620  148.22226 1.0229793
## alpha[37] 0.8954617 1.090488  224.56484 1.0148189
## alpha[38] 1.2264140 1.488497  137.67816 1.0412943
```

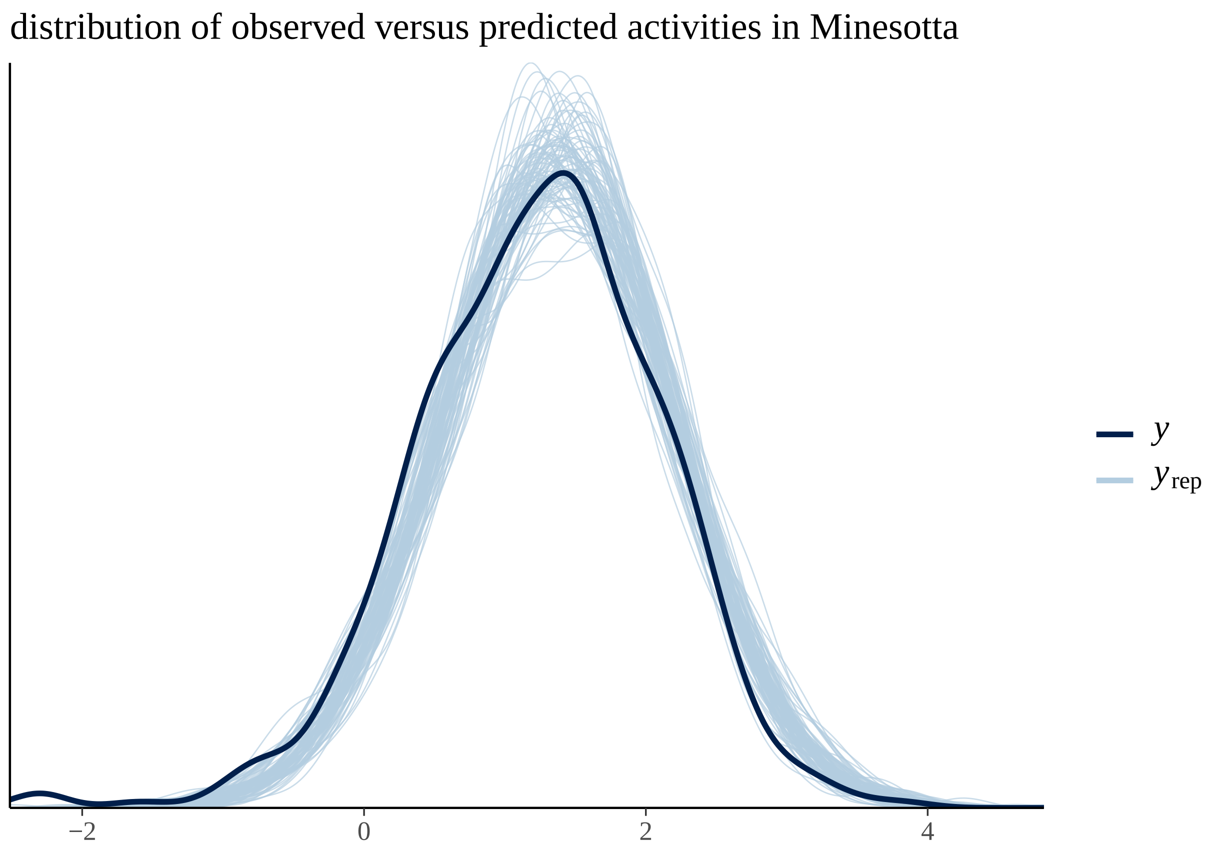
```
## alpha[39] 1.7487282 1.956366  643.76925 0.9967498
## alpha[40] 1.9885371 2.240714  375.38365 1.0011813
## alpha[41] 1.9179766 2.145704  706.54201 0.9970277
## alpha[42] 1.6710101 1.904157  516.73995 0.9995652
## alpha[43] 1.6055740 1.803004  615.38856 1.0003689
## alpha[44] 1.5640274 1.727169  255.33988 1.0114678
## alpha[45] 1.5564288 1.731487  379.44091 1.0097973
## alpha[46] 1.5424968 1.699106  482.36256 1.0094027
## alpha[47] 1.3847398 1.588140  466.66873 1.0008019
## alpha[48] 1.4008471 1.591971  691.35284 1.0120459
## alpha[49] 1.7575411 1.940129  694.97242 0.9997251
## alpha[50] 1.9196443 2.181785  382.69595 0.9954617
## alpha[51] 1.8525745 2.094359  311.22759 1.0002242
## alpha[52] 1.9080005 2.097817  344.49253 1.0048848
## alpha[53] 1.7041547 1.915230  622.60646 1.0079671
## alpha[54] 1.5556057 1.687559  275.40253 1.0229144
## alpha[55] 1.4929181 1.732322  339.51760 0.9972675
## alpha[56] 1.4723632 1.700404  612.64026 0.9977644
## alpha[57] 1.3140267 1.478220  191.97718 1.0281935
## alpha[58] 1.9362344 2.207742  447.89250 0.9969725
## alpha[59] 1.7703608 2.052669  518.15364 0.9989055
## alpha[60] 1.7603745 2.014484  468.91846 1.0055058
## alpha[61] 1.2220093 1.376650 1281.40295 0.9941051
## alpha[62] 1.8928019 2.101918  447.51001 0.9933321
## alpha[63] 1.8520489 2.118180  342.72631 1.0074008
## alpha[64] 1.7838353 1.993645  930.45855 0.9978107
## alpha[65] 1.9106162 2.176787  277.08797 1.0230820
## alpha[66] 1.5436192 1.754322  345.34346 1.0087622
## alpha[67] 1.6943072 1.917247  485.74552 1.0008180
## alpha[68] 1.1023889 1.316839  675.01776 0.9976346
## alpha[69] 1.6760889 1.904531  410.54510 1.0143625
## alpha[70] 0.9467748 1.031544  460.56142 1.0122695
## alpha[71] 1.5925697 1.722826  804.23012 0.9980156
## alpha[72] 1.7403166 1.911711 1033.51353 0.9953939
## alpha[73] 1.9212688 2.156548  426.31515 1.0038973
## alpha[74] 1.7026229 1.860355  155.84069 1.0466542
## alpha[75] 1.5745293 1.794677  567.35268 1.0003115
## alpha[76] 1.9671360 2.181727  476.34349 0.9977621
## alpha[77] 1.7210890 1.967809  643.86509 0.9991457
## alpha[78] 1.1534945 1.347036  424.80128 1.0076298
## alpha[79] 1.5721378 1.784654  193.65638 1.0387235
## alpha[80] 1.3967809 1.514704  824.09018 0.9968114
## alpha[81] 1.8627922 2.102567  333.86921 1.0003294
## alpha[82] 1.7951899 2.060394  314.66094 0.9966839
## alpha[83] 1.8270149 2.025605  600.21807 1.0056050
## alpha[84] 1.5713577 1.756934  625.98363 0.9981667
## alpha[85] 1.7914344 2.020523  265.54125 1.0091448
## gamma0     1.4929401 1.554482  192.37246 1.0070651
## gamma1     0.7962812 0.926071  161.57319 1.0123554
```

```r
summary(mod2)$summary[c("gamma0", "gamma1"),]
```

```
##             mean     se_mean         sd      2.5%       25%      50%       75%
## gamma0 1.4693273 0.002956885 0.04101152 1.3913669 1.4423267 1.467719 1.4929401
## gamma1 0.7311968 0.007691869 0.09777246 0.5387533 0.6611365 0.734300 0.7962812
```

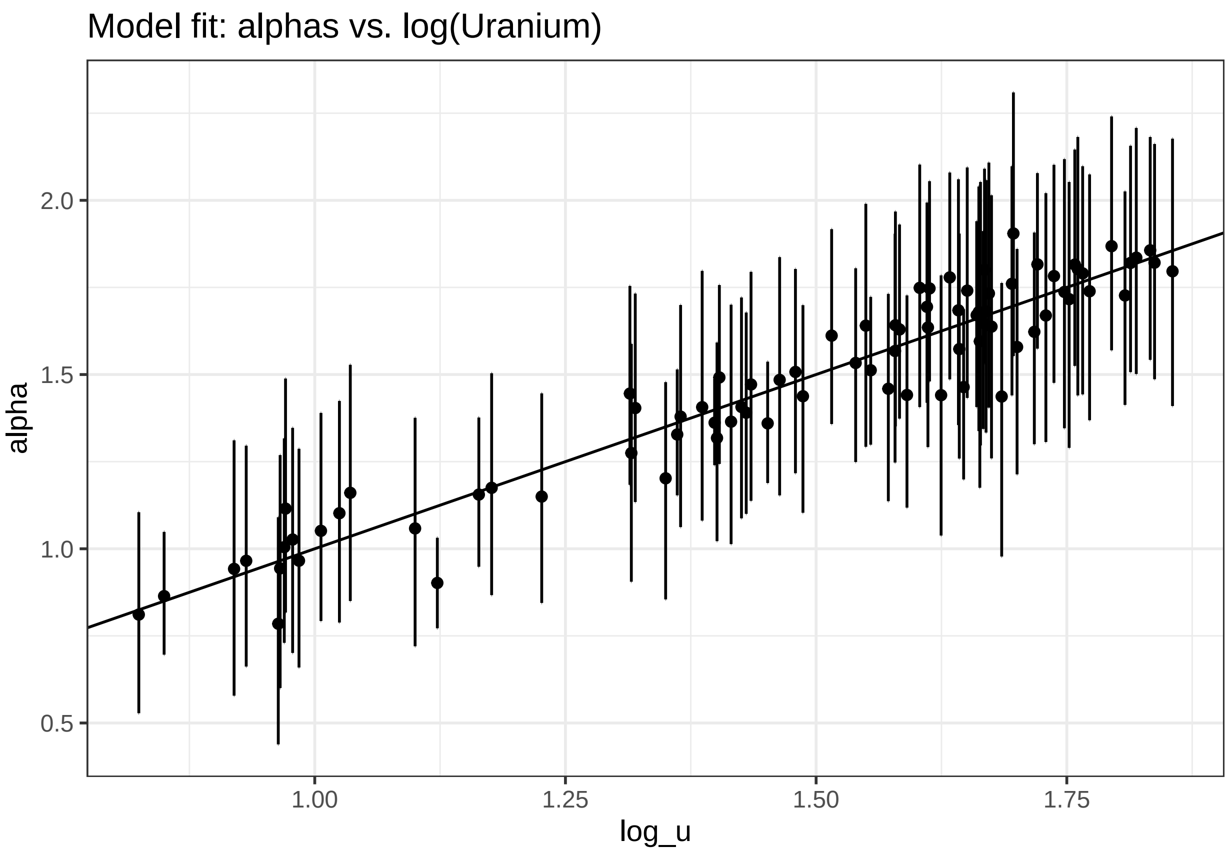
```
##           97.5%    n_eff     Rhat
## gamma0 1.554482 192.3725 1.007065
## gamma1 0.926071 161.5732 1.012355
```

```
# Plot the density and simulations to check if everthing is Ok
y <- log(dmn$activity)
yrep2 <- extract(mod2)[["y_rep"]]
samp100 <- sample(nrow(yrep2), 100)
ppc_dens_overlay(y, yrep2[samp100,])  + ggtitle("distribution of observed versus predicted activities in
```




```
#4. For $\alpha$ plot, get median estimates of alpha's, and the 2.5th and 97.5th percentiles. Also get

## Generating the graph pg 43

# Collecting Data
alphas <- summary(mod2)$summary[c(paste0("alpha[", 1:85, "]")),]
gamma <- summary(mod2)$summary[c("gamma0", "gamma1"),]

# preparing data.frame
lowalpha <- alphas[,"2.5%"]
upalpha <- alphas[,"97.5%"]
log_u <- gamma[1,"mean"] + gamma[2,"mean"]*log(Ncty$Uppm)

dgp43 <- as.tibble(data.frame(alpha = alphas[,"mean"],
                              lowalpha = lowalpha,
                              upalpha = upalpha,
                              log_u = log_u))
```

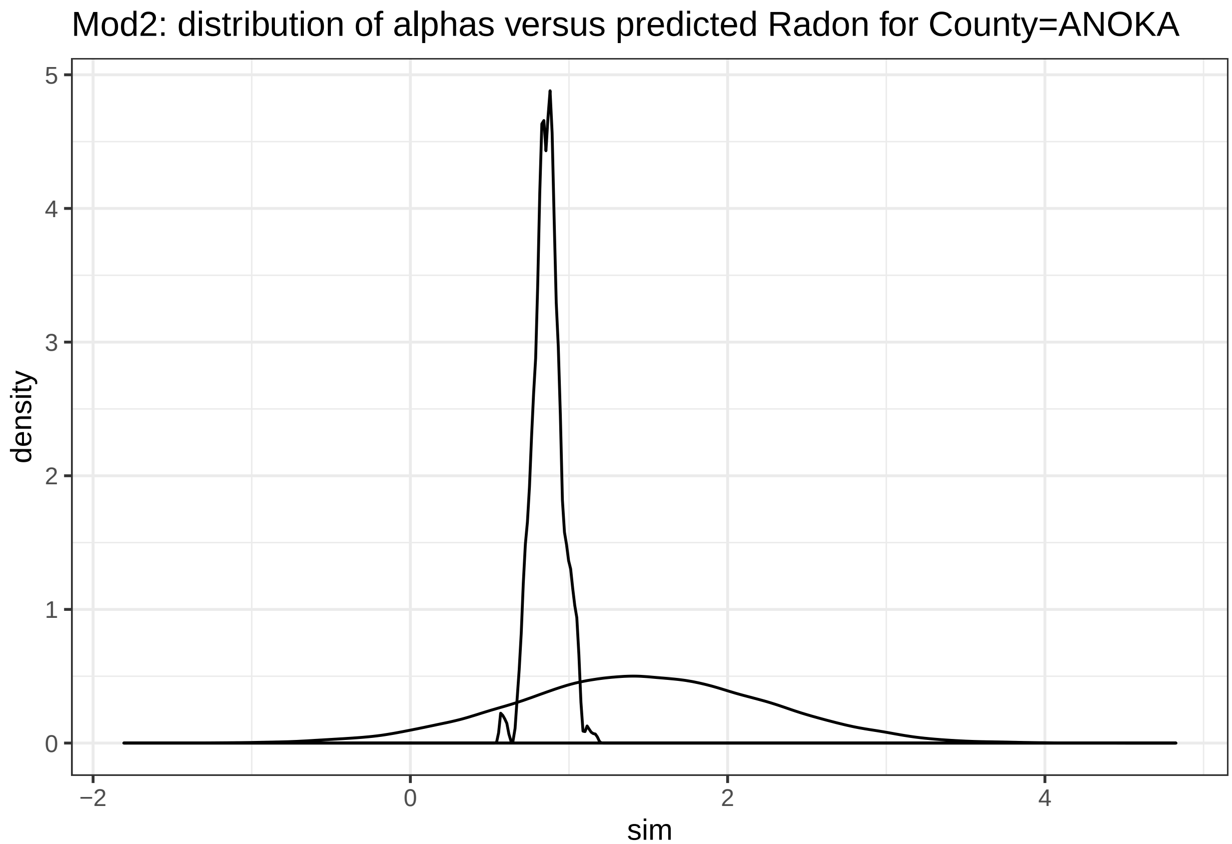
```
# Plotting the grapg - pg43
dgp43 %>%
  ggplot(aes(x = log_u, y = alpha)) +
  geom_errorbar(aes(ymin=lowalpha, ymax=upalpha)) +
  geom_point() + geom_abline() +
  ggtitle("Model fit: alphas vs. log(Uranium)") +
  theme_bw()
```




```
#5. For the predicted y plot, you will need your posterior predictive samples for $y$'s and then just u

y_rep   <- extract(mod2)[["y_rep"]][,5:56]          # getting just simulations of county = "ANOKA"
alphaANOKA <- extract(mod2)[["alpha"]][,2]   # getting the alphas of county "ANOKA"

dgp32 <- as.tibble(data.frame(sim = 1:500,
                              alpha = alphaANOKA,
                              y_rep = y_rep))

dgp32 <- dgp32 %>%
  pivot_longer("y_rep.1":"y_rep.52", values_to = "y_rep")
```

```
# Plotting the grapg - pg32
dgp32 %>%
  ggplot(aes(x = sim)) +
  geom_density(aes(alpha), alpha = 0.1) +
  geom_density(aes(y_rep), alpha = 0.1) +
  ggtitle("Mod2: distribution of alphas versus predicted Radon for County=ANOKA") +
```

```
theme_bw()
```


Mod2: distribution of alphas versus predicted Radon for County=ANOKA
density
5
4
3
2
1
0
−2
0
2
4
sim