Applied Bayesian Modeling - modules 7 and 8

Leontine Alkema

September 25, 2022

1 Read in radon data

Read in the radon data and process (copied from earlier module)

```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 (county) 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, sep = ",")
cty <-
  cty %>%
  mutate(fips = 1000 * stfips + ctfips) %>%
  dplyr::select(fips, Uppm) %>%
  rename(ura_county = (Uppm))

dmn <- d %>%
  filter(state=="MN") %>% # Minnesota data only
  dplyr::select(fips, county, floor, activity) %>%
  left_join(cty)
```

1.1 More data processing for multilevel modeling

Some more data processing first, to produce a data set that has county info and to produce the plot with means from the slides. In the data set:

- y_i is log(activity)
- county gives county name (fips gives the unique county ID)
- x_i is floor
- u_i is log_ur = log(ura_county)

(the last two are added for module 8, when including predictors)

```r
dat <-
  dmn%>%
  mutate(y = log(activity), log_ur = log(ura_county))
head(dat)
```

```
##    fips               county floor activity ura_county         y     log_ur
## 1 27001 AITKIN                   1      2.2   0.502054 0.7884574 -0.6890476
## 2 27001 AITKIN                   0      2.2   0.502054 0.7884574 -0.6890476
## 3 27001 AITKIN                   0      2.9   0.502054 1.0647107 -0.6890476
## 4 27001 AITKIN                   0      1.0   0.502054 0.0000000 -0.6890476
## 5 27003 ANOKA                    0      3.1   0.428565 1.1314021 -0.8473129
## 6 27003 ANOKA                    0      2.5   0.428565 0.9162907 -0.8473129
```

Create summary data set with info for each county:

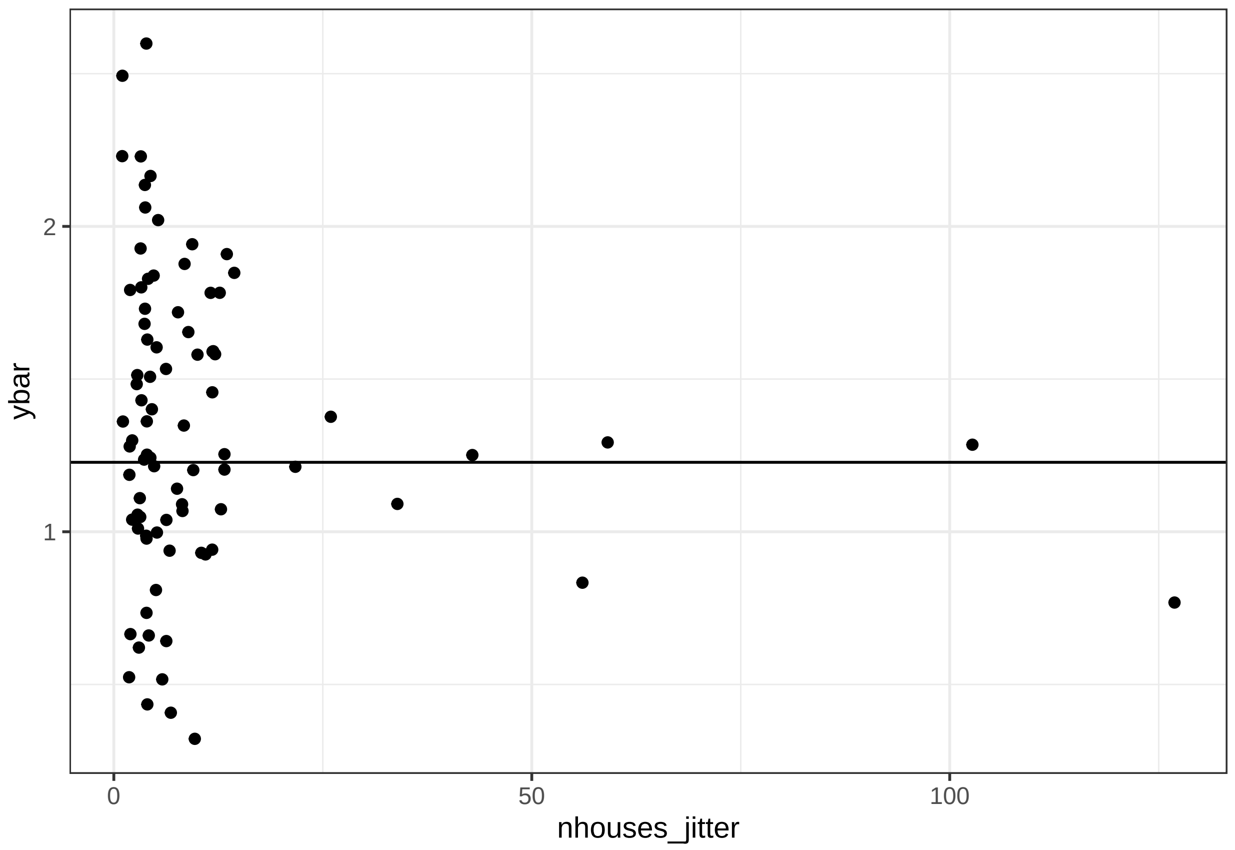
```r
# to plot observations and county means ~ sample sizes,
# easier to see if sample sizes are slighly jittered
set.seed(12345)

datcounty <- dat %>%
  group_by(fips) %>%
  summarize(nhouses = n(), ybar = mean(y), county = county[1], log_ur = log_ur[1]) %>%
  mutate(nhouses_jitter = nhouses*exp(runif (length(nhouses), -.1, .1)))
ngroups <- dim(datcounty)[1]
head(datcounty)
```

```
## # A tibble: 6 x 6
##    fips nhouses  ybar county                 log_ur nhouses_jitter
##   <dbl>   <int> <dbl> <chr>                   <dbl>          <dbl>
## 1 27001       4 0.660 "AITKIN              " -0.689           4.18
## 2 27003      52 0.833 "ANOKA               " -0.847          56.1
## 3 27005       3 1.05  "BECKER              " -0.113           3.16
## 4 27007       7 1.14  "BELTRAMI            " -0.593           7.56
## 5 27009       4 1.25  "BENTON              " -0.143           3.97
## 6 27011       3 1.51  "BIG STONE           "  0.387           2.81
```

```r
ybarbar <- mean(dat$y) # population (here state) mean
```

```r
datcounty %>%
  ggplot(aes(x = nhouses_jitter, y = ybar)) +
  geom_point() +
  geom_hline(mapping = aes(yintercept = ybarbar)) +
  theme_bw()
```



2 Model fitting w/o predictors

```
fit <- brm(y ~ (1|county),
        data = dat,
        file = "output/mod7.8ex2",
        iter = 1000,
        chains = 4,
        cores = getOption("mc.cores", 4))
```

Summary of model fit:

```
summary(fit)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ (1 | county)
##    Data: dat (Number of observations: 927)
##   Draws: 4 chains, each with iter = 1000; warmup = 500; thin = 1;
##          total post-warmup draws = 2000
##
## Group-Level Effects:
## ~county (Number of levels: 85)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.32      0.05     0.23     0.43 1.01      609     1187
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.32      0.05     1.22     1.42 1.00     1629     1753
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.81      0.02     0.77     0.85 1.00     3576     1497
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

2.1 Visualizing the group-level mean parameters

Coefficients can be obtained using coef(fit), you can get the help file here:

```
#?coef.brmsfit
```

Just showing some function calls here first, ie for mu_alpha:

```
fixef(fit)
```

```
##           Estimate  Est.Error     Q2.5    Q97.5
## Intercept 1.317609 0.05099501 1.218875 1.418319
```

eta = alpha - mu_alpha (as compared to notation in slides), labeled here as random effects

```
eta <- as_tibble(ranef(fit)$county[,,"Intercept"], rownames = "county")
head(eta)
```

```
## # A tibble: 6 x 5
##   county                  Estimate Est.Error   Q2.5  Q97.5
##   <chr>                      <dbl>     <dbl>  <dbl>  <dbl>
## 1 "AITKIN              "   -0.254      0.253 -0.773  0.218
## 2 "ANOKA               "   -0.428      0.111 -0.651 -0.213
## 3 "BECKER              "   -0.0919     0.263 -0.603  0.426
## 4 "BELTRAMI            "   -0.0896     0.225 -0.537  0.331
## 5 "BENTON              "   -0.0240     0.241 -0.496  0.467
## 6 "BIG STONE           "    0.0657     0.260 -0.433  0.570
```

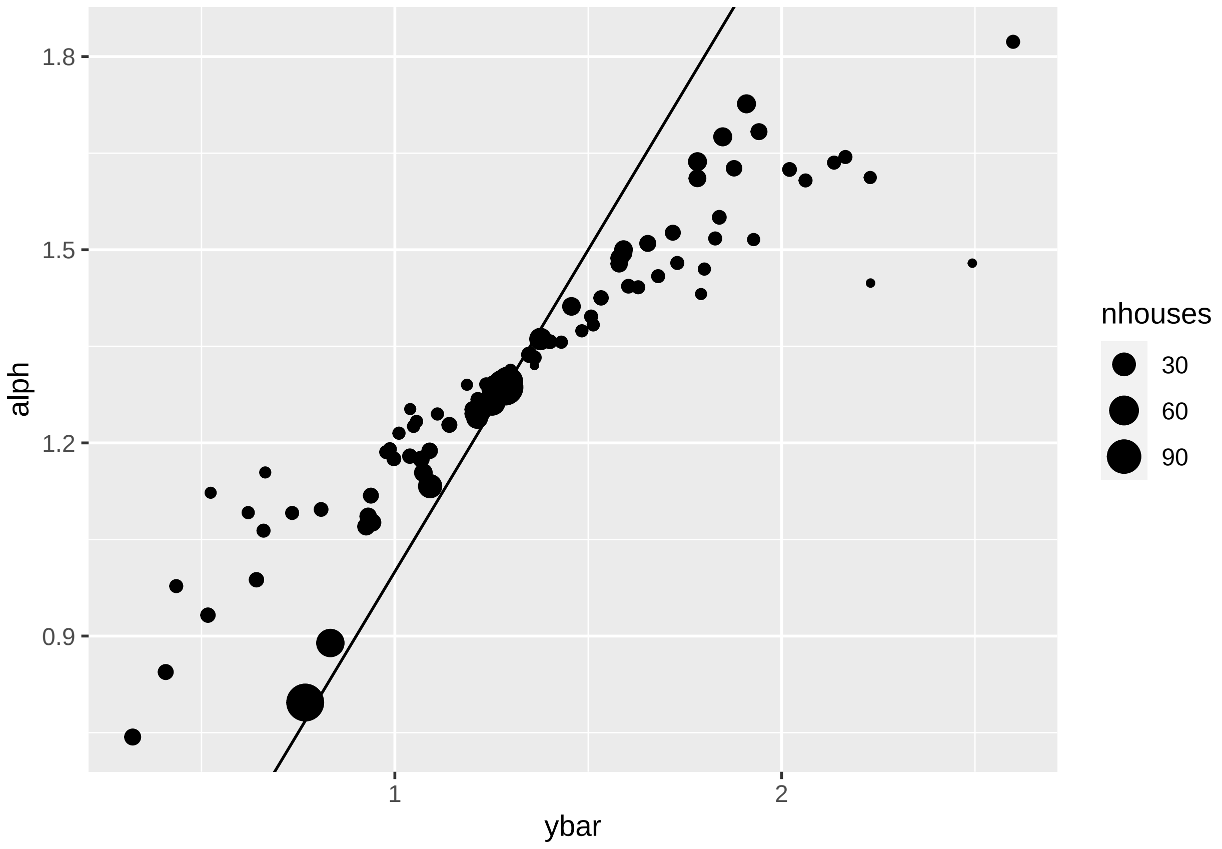
To get the alpha = eta + mu_alpha, we can use the following call

```
alphas <-
  coef(fit, summary = T)$county %>%
  as_tibble(rownames = "county") %>%
  rename(alph = Estimate.Intercept)
alphas
```

```
## # A tibble: 85 x 5
##    county                   alph Est.Error.Intercept Q2.5.Intercept Q97.5.Inter~1
##    <chr>                   <dbl>               <dbl>          <dbl>         <dbl>
##  1 "AITKIN               " 1.06                0.253          0.520          1.54
##  2 "ANOKA                " 0.889               0.101          0.697          1.08
##  3 "BECKER               " 1.23                0.265          0.724          1.74
##  4 "BELTRAMI             " 1.23                0.224          0.782          1.66
##  5 "BENTON               " 1.29                0.243          0.805          1.77
##  6 "BIG STONE            " 1.38                0.261          0.863          1.89
##  7 "BLUE EARTH           " 1.73                0.191          1.35           2.09
##  8 "BROWN                " 1.44                0.257          0.938          1.96
##  9 "CARLTON              " 1.09                0.190          0.693          1.45
## 10 "CARVER               " 1.25                0.195          0.858          1.63
## # ... with 75 more rows, and abbreviated variable name 1: Q97.5.Intercept
```

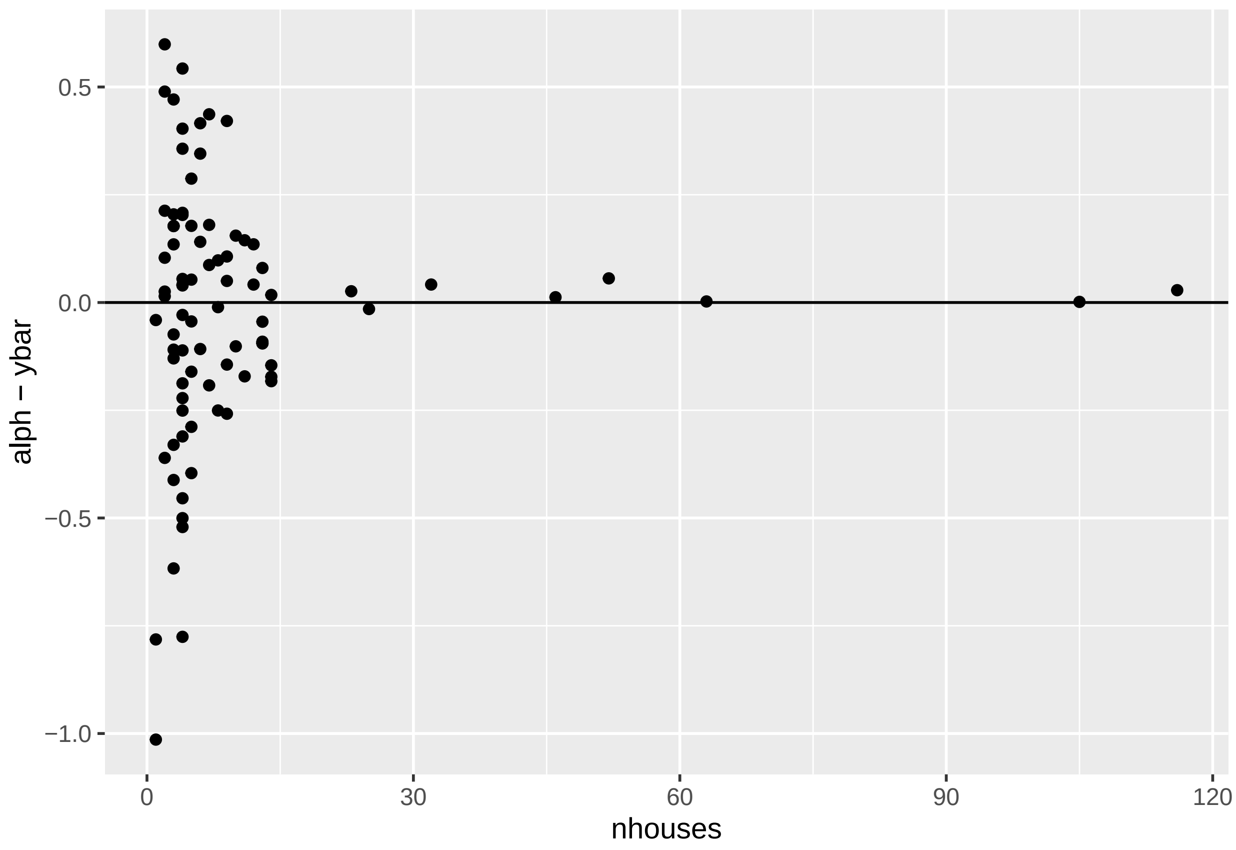
Make the plot of alpha ~ ybar

```
alphas %>%
  left_join(datcounty) %>%
  ggplot(aes(y = alph, x = ybar, size = nhouses)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0)
```



Plot of alpha - ybar

```
alphas %>%
  left_join(datcounty) %>%
  ggplot(aes(y = alph - ybar, x = nhouses)) +
  geom_point() +
  geom_hline(yintercept = 0)
```



2.2 Predicting radon levels in unsampled houses and unsampled counties

2.2.1 Do-it-yourself sampling based approach

We first obtain posterior samples of the hyperparameters and then sample from the predictive distribution for our outcomes of interest, e.g. new observations or group means for new groups.

Extract the posterior samples from the brm-fit:

```
samp <- as_draws_df(fit)
#dim(samp)
#names(samp)[1:3]
sigmay_s <- samp$sigma
mualpha_s <- samp$b_Intercept
sigmaalpha_s <- samp$sd_county__Intercept
S <- length(sigmay_s)
county_names <- rownames(ranef(fit)$county)
```

Sampling for a new observation, for example from 1st county Aitkin:

```r
county_names[1] # just note there are spaces in names, so little more annoying to work with
```

```
## [1] "AITKIN              "
```

```r
alpha1_s <- mualpha_s + samp$`r_county[AITKIN..............,Intercept]`

set.seed(1234) # to make the sampling reproducible
ytilde_s <- rnorm(S, alpha1_s, sigmay_s)
```

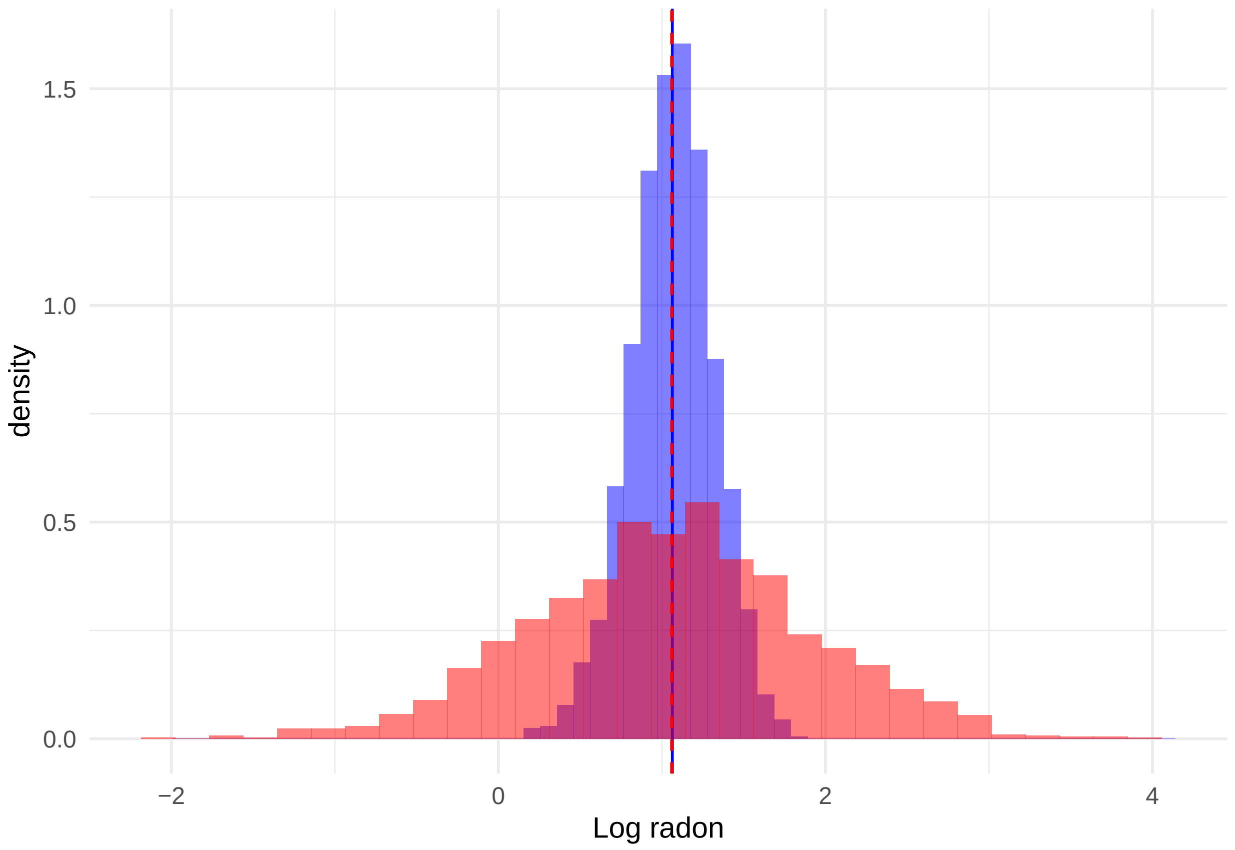
Point estimates and 95% CI (using point_interval function from tidybayes package here):

```r
point_interval(ytilde_s, .point = mean)
```

```
##          y       ymin     ymax .width .point .interval
## 1 1.058922 -0.5590669 2.695411   0.95   mean        qi
```

Visualize densities

```r
p <- as_tibble(alpha1_s) %>%
  ggplot(aes(alpha1_s, after_stat(density))) +
  geom_histogram(alpha = .5, fill = "blue", bins = 60) +
  theme_minimal() +
  xlab("Log radon") +
  geom_vline(xintercept = mean(alpha1_s), col = "blue")
p +
  geom_histogram(as_tibble(ytilde_s), bins = 30, mapping = aes(ytilde_s, after_stat(density)),
               alpha = .5, fill = "red", adjust = 1.5, size = 1.5) +
  geom_vline(xintercept = mean(ytilde_s), col = "red", linetype = "dashed")
```

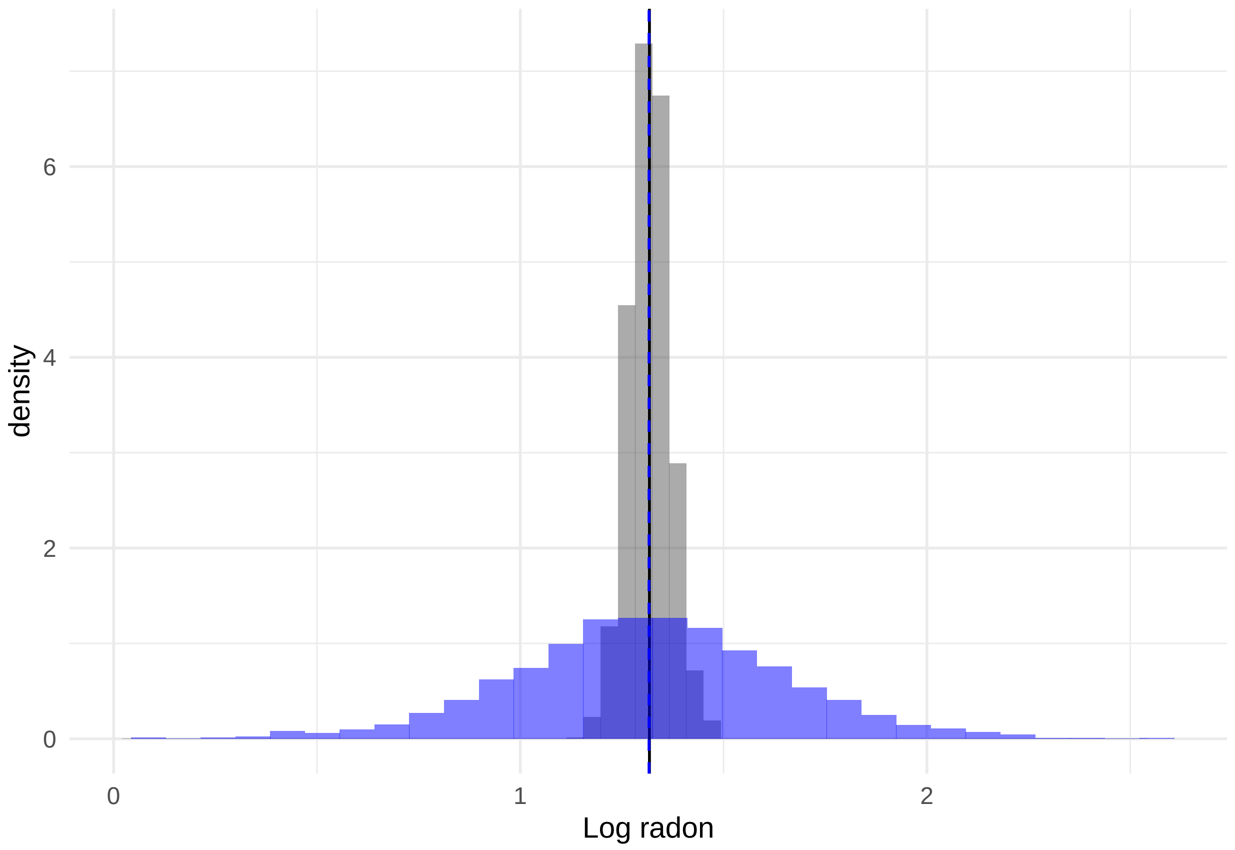


Sampling for a new group mean:

```r
set.seed(1234) # to make the sampling reproducible
alphanew_s <- rnorm(length(sigmaalpha_s), mualpha_s, sigmaalpha_s)
```

Visualizing the densities

```r
p <- as_tibble(mualpha_s) %>%
  ggplot(aes(mualpha_s, after_stat(density))) +
  geom_histogram(alpha = .5, bins = 60) +
  theme_minimal() +
  xlab("Log radon") +
  geom_vline(xintercept = mean(mualpha_s))
p +
  geom_histogram(aes(alphanew_s, after_stat(density)), alpha = .5, fill = "blue", bins = 30, fill = "blu
  geom_vline(xintercept = mean(alphanew_s), col = "blue", linetype = "dashed")
```



2.2.2 Can brm-functions do this for me?

Yes, they can! And when you've fully understood what you're doing, I recommend you take this approach :)

Prediction for a new house in county 1, we just need a data frame with the county name here (given no predictors)

```
newdata1 <- data.frame(
  county = county_names[1]
)
ytilde_brm_s <- posterior_predict(fit, newdata = newdata1)
```

Compare the two intervals, should be approximately the same

```
point_interval(ytilde_s, .point = mean)
```

```
##          y       ymin     ymax .width .point .interval
## 1 1.058922 -0.5590669 2.695411   0.95   mean        qi
```

```
point_interval(ytilde_brm_s, .point = mean)
```

```
##          y       ymin     ymax .width .point .interval
## 1 1.071703 -0.5788478 2.702977   0.95   mean        qi
```

3 Model fitting with predictors

Add group-level predictor floor:

```
fit2 <- brm(y ~ (1+floor|county) + floor,
            file = "output/mod7.8ex3",
            data = dat, sample_prior = T, chains = 4,
            iter = 2000, thin = 1,
            cores = getOption("mc.cores", 4))
```

```
summary(fit2)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ (1 + floor | county) + floor
##    Data: dat (Number of observations: 927)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~county (Number of levels: 85)
##                      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)            0.35      0.05     0.26     0.46 1.00     1129
## sd(floor)                0.27      0.15     0.02     0.56 1.01      559
## cor(Intercept,floor)    -0.19      0.39    -0.87     0.75 1.00     2210
##                      Tail_ESS
## sd(Intercept)            2211
## sd(floor)                1261
## cor(Intercept,floor)     1482
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.47      0.06     1.36     1.58 1.00     1544     2018
## floor        -0.68      0.09    -0.85    -0.51 1.00     2603     2543
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.76      0.02     0.73     0.80 1.00     3175     2914
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

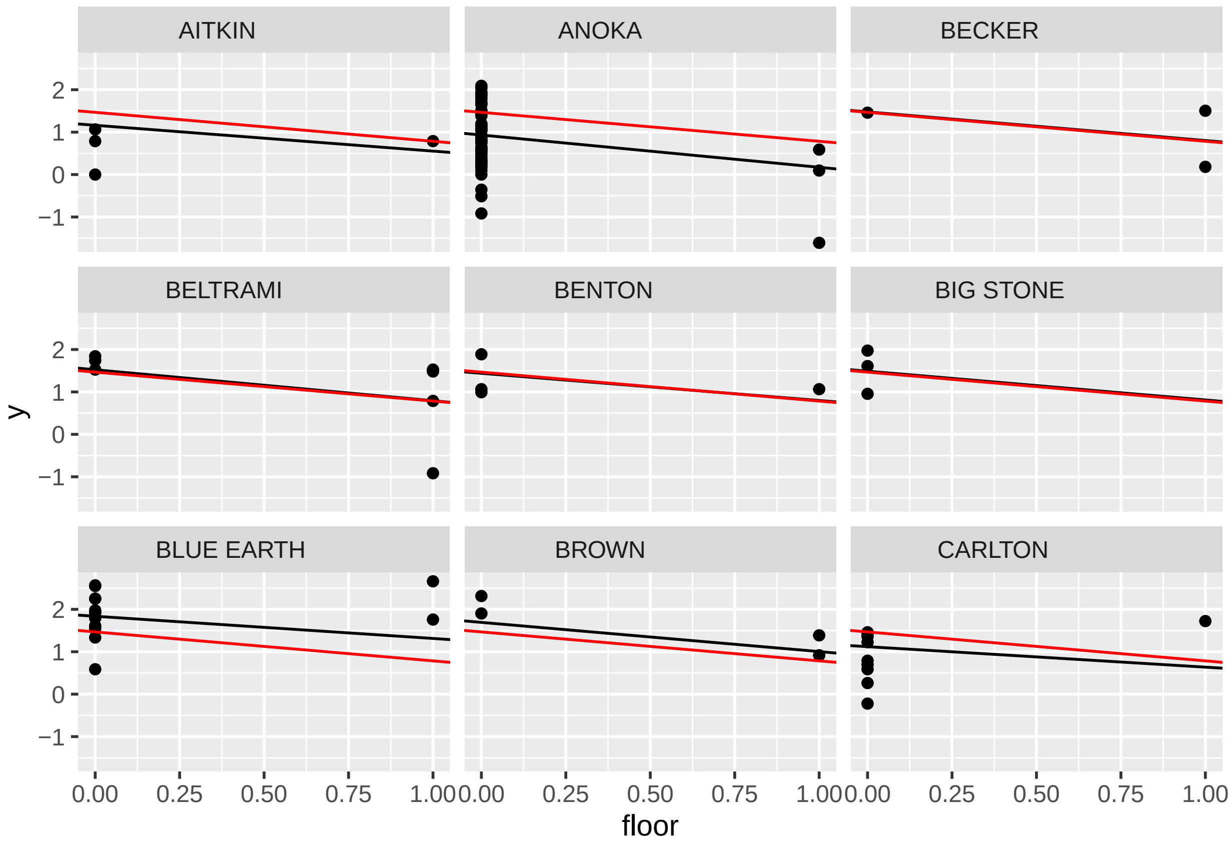
Visualize the fitted regression line (although here just two values for the covariate x, 0 and 1), for each county

```
coefs <- coef(fit2)$county[, 'Estimate', c("Intercept", "floor")]

coefs_tibble <- as_tibble(rownames =  "county", coefs) %>%
  rename(slope = floor)

dat %>%
```

```r
  full_join(coefs_tibble, by = "county") %>%
  filter(county %in% coefs_tibble$county[1:9]) %>% # just select 4 counties
  ggplot(aes(x = floor, y = y)) +
  geom_point() +
  geom_abline(aes(intercept = Intercept, slope = slope)) +
  geom_abline(aes(intercept = fixef(fit2)[, "Estimate"][1],
                  slope = fixef(fit2)[, "Estimate"][2]), col = "red") +
  facet_wrap( ~ county)
```



Fit the full model, including county-level uranium

```r
fit3 <- brm(y ~ (1 + floor|county) + log_ur*floor, family = gaussian(),
            file = "output/mod7.8ex3.2",
            data = dat, sample_prior = T, chains = 4,
            iter = 2000, thin = 1,
            cores = getOption("mc.cores", 4))
```

```r
summary(fit3)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ (1 + floor | county) + log_ur * floor
##    Data: dat (Number of observations: 927)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
```

```
## Group-Level Effects:
## ~county (Number of levels: 85)
##                      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)            0.14      0.05     0.03     0.24 1.00      943
## sd(floor)                0.31      0.13     0.04     0.57 1.01      841
## cor(Intercept,floor)     0.28      0.43    -0.64     0.96 1.01      922
##                      Tail_ESS
## sd(Intercept)             990
## sd(floor)                 925
## cor(Intercept,floor)     1261
##
## Population-Level Effects:
##              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept        1.47      0.04     1.40     1.55 1.00     3112     3456
## log_ur           0.81      0.10     0.62     1.01 1.00     3890     2716
## floor           -0.67      0.09    -0.84    -0.47 1.00     3736     2653
## log_ur:floor    -0.40      0.23    -0.85     0.06 1.00     4034     3044
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.76      0.02     0.73     0.80 1.00     4571     2990
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```