Full applications w/Bayes

Plus some ways of handling missing data

Agenda

- Equation practice
- Fitting multilevel logistic regression models with **brms**
    - Applied walkthrough 1: Twitter data
    - Applied walkthrough 2: Lung cancer data
- Missing values

Equation practice

Data

Read in the `nurses.csv` data. Note – each row represents data for one nurse.

```
library(tidyverse)
nurses <- read_csv(here::here("data", "nurses.csv"))
```

02:00

Data

```
nurses
```

```
## # A tibble: 1,000 x 11
##    hospital  ward wardid nurse   age gender experien stress wardtype
##       <dbl> <dbl>  <dbl> <dbl> <dbl> <chr>     <dbl>  <dbl> <chr>
##  1        1     1     11     1    36 Male         11      7 general care
##  2        1     1     11     2    45 Male         20      7 general care
##  3        1     1     11     3    32 Male          7      7 general care
##  4        1     1     11     4    57 Female       25      6 general care
##  5        1     1     11     5    46 Female       22      6 general care
##  6        1     1     11     6    60 Female       22      6 general care
##  7        1     1     11     7    23 Female       13      6 general care
##  8        1     1     11     8    32 Female       13      7 general care
##  9        1     1     11     9    60 Male         17      7 general care
## 10        1     2     12    10    45 Male         21      6 special care
## # … with 990 more rows
```

Model 1

Fit the following model

$$\text{stress}_i \sim N\left(\alpha_{j[i]} + \beta_{1j[i]}(\text{experien}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{wardtype}_{\text{special care}}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{wardtype}_{\text{special care}}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for wardid j} = 1, \ldots, \text{J}$$

```
lmer(stress ~ experien * wardtype + (experien|wardid),
     data = nurses)

# or
lmer(stress ~ experien + wardtype + experien:wardtype +
       (experien|wardid),
     data = nurses)
```

02:00

Model 2

Fit the following model

$$\text{stress}_i \sim N\left(\alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{experien}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{wardtype}_{\text{special care}}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{wardtype}_{\text{special care}}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for wardid j} = 1, \ldots, \text{J}$$

$$\begin{pmatrix} \alpha_k \\ \beta_{1k} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{hospsize}_{\text{medium}}) + \gamma_2^{\alpha}(\text{hospsize}_{\text{small}}) \\ \mu_{\beta_{1k}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_k} & \rho_{\alpha_k\beta_{1k}} \\ \rho_{\beta_{1k}\alpha_k} & \sigma^2_{\beta_{1k}} \end{pmatrix}\right), \text{for hospital k} = 1, \ldots, \text{K}$$

```
lmer(stress ~ experien * wardtype + hospsize +
       (experien|wardid) + (experien|hospital),
     data = nurses)
```

02:00

Model 3

Fit the following model

$$\text{stress}_i \sim N\left(\alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{experien}) + \beta_2(\text{age}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{expcon}_{\text{experiment}}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & 0 \\ 0 & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for wardid j} = 1, \ldots, \text{J}$$

$$\begin{pmatrix} \alpha_k \\ \beta_{1k} \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_{\alpha_k} \\ \mu_{\beta_{1k}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_k} & 0 \\ 0 & \sigma^2_{\beta_{1k}} \end{pmatrix}\right), \text{for hospital k} = 1, \ldots, \text{K}$$

```
lmer(stress ~ experien + age + expcon +
       (experien||wardid) + (experien||hospital),
     data = nurses)

# or

lmer(stress ~ experien + age + expcon +
       (1|wardid) + (0 + experien|wardid) +
       (1|hospital) + (0 + experien|hospital),
     data = nurses)
```

Model 4

Fit the following model

$$\text{stress}_i \sim N\left(\alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{experien}) + \beta_2(\text{age}), \sigma^2\right)$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{expcon}_{\text{experiment}}) + \gamma_2^\alpha(\text{wardtype}_{\text{special care}}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for wardid j} = 1, \ldots, \text{J}$$

$$\begin{pmatrix} \alpha_k \\ \beta_{1k} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{hospsize}_{\text{medium}}) + \gamma_2^\alpha(\text{hospsize}_{\text{small}}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{hospsize}_{\text{medium}}) + \gamma_2^{\beta_1}(\text{hospsize}_{\text{small}}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_k} & \rho_{\alpha_k\beta_{1k}} \\ \rho_{\beta_{1k}\alpha_k} & \sigma^2_{\beta_{1k}} \end{pmatrix}\right), \text{for hospital k} = 1, \ldots, \text{K}$$

```
lmer(stress ~ experien * hospsize + age + expcon + wardtype +
       (experien|wardid) + (experien|hospital),
     data = nurses)
```

02:00

Model 5

Fit the following model

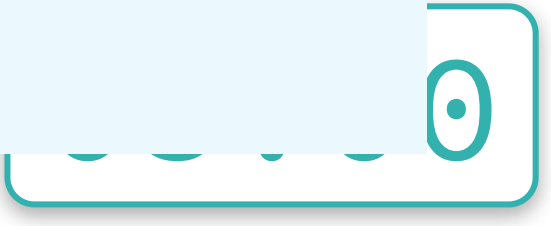
$$\text{expcon}_i \sim \text{Binomial}(n = 1, \text{prob}_{\text{expcon=experiment}} = \widehat{P})$$

$$\log\left[\frac{\hat{P}}{1-\hat{P}}\right] = \alpha_{j[i],k[i]} + \beta_{1j[i]}(\text{age})$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_{\alpha_j} \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for wardid j} = 1, \ldots, \text{J}$$

$$\alpha_k \sim N\left(\mu_{\alpha_k}, \sigma^2_{\alpha_k}\right), \text{for hospital k} = 1, \ldots, \text{K}$$

```
nurses <- nurses %>%
  mutate(expcon = factor(expcon))

glmer(expcon ~ age +
        (age|wardid) + (1|hospital),
      data = nurses,
      family = binomial(link = "logit"),
      data = nurses)
```



Model 6

Fit the following model

$$\text{expcon}_i \sim \text{Binomial}(n = 1, \text{prob}_{\text{expcon=experiment}} = \widehat{P})$$

$$\log\left[\frac{\hat{P}}{1-\hat{P}}\right] = \alpha_{j[i],k[i]} + \beta_{1j[i]}(\text{age})$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{wardtype}_{\text{special care}}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for wardid j} = 1, \ldots, \text{J}$$

$$\alpha_k \sim N\left(\gamma_0^\alpha + \gamma_1^\alpha(\text{hospsize}_{\text{medium}}) + \gamma_2^\alpha(\text{hospsize}_{\text{small}}), \sigma^2_{\alpha_k}\right), \text{for hospital k} = 1, \ldots, \text{K}$$

```
glmer(expcon ~ hospsize + age + wardtype +
        (age|wardid) + (1|hospital),
      data = nurses,
      family = binomial(link = "logit"))
```

02:00

Model 7

Fit the following model

$$\text{expcon}_i \sim \text{Binomial}(n = 1, \text{prob}_{\text{expcon=experiment}} = \widehat{P})$$

$$\log\left[\frac{\hat{P}}{1-\hat{P}}\right] = \alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{age}) + \beta_2(\text{gender}_{\text{Male}}) + \beta_3(\text{age} \times \text{gender}_{\text{Male}})$$

$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{wardtype}_{\text{special care}}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma^2_{\beta_{1j}} \end{pmatrix}\right), \text{for wardid j} = 1, \ldots, \text{J}$$

$$\begin{pmatrix} \alpha_k \\ \beta_{1k} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{hospsize}_{\text{medium}}) + \gamma_2^\alpha(\text{hospsize}_{\text{small}}) \\ \mu_{\beta_{1k}} \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_k} & \rho_{\alpha_k\beta_{1k}} \\ \rho_{\beta_{1k}\alpha_k} & \sigma^2_{\beta_{1k}} \end{pmatrix}\right), \text{for hospital k} = 1, \ldots, \text{K}$$

```
glmer(expcon ~ age * gender + wardtype + hospsize +
        (age|wardid) + (age|hospital),
      data = nurses,
      family = binomial(link = "logit"))
```

02:00

Model 8

Fit the following model

$$\text{expcon}_i \sim \text{Binomial}(n = 1, \text{prob}_{\text{expcon=experiment}} = \widehat{P})$$

$$\log\left[\frac{\hat{P}}{1-\hat{P}}\right] = \alpha_{j[i],k[i]} + \beta_1(\text{age}) + \beta_2(\text{gender}_{\text{Male}}) + \beta_{3j[i],k[i]}(\text{experien})$$

$$\begin{pmatrix} \alpha_j \\ \beta_{3j} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{wardtype}_{\text{special care}}) \\ \gamma_0^{\beta_3} + \gamma_1^{\beta_3}(\text{wardtype}_{\text{special care}}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_j} & \rho_{\alpha_j\beta_{3j}} \\ \rho_{\beta_{3j}\alpha_j} & \sigma^2_{\beta_{3j}} \end{pmatrix}\right), \text{for wardid j} = 1, \ldots, \text{J}$$

$$\begin{pmatrix} \alpha_k \\ \beta_{3k} \end{pmatrix} \sim N\left(\begin{pmatrix} \gamma_0^{\alpha} + \gamma_1^{\alpha}(\text{hospsize}_{\text{medium}}) + \gamma_2^{\alpha}(\text{hospsize}_{\text{small}}) \\ \gamma_0^{\beta_3} + \gamma_1^{\beta_3}(\text{hospsize}_{\text{medium}}) + \gamma_2^{\beta_3}(\text{hospsize}_{\text{small}}) \end{pmatrix}, \begin{pmatrix} \sigma^2_{\alpha_k} & \rho_{\alpha_k\beta_{3k}} \\ \rho_{\beta_{3k}\alpha_k} & \sigma^2_{\beta_{3k}} \end{pmatrix}\right), \text{for hospital k} = 1, \ldots, \text{K}$$

```
glmer(expcon ~ age + gender +
        experien * wardtype + experien * hospsize +
        (experien|wardid) + (experien|hospital),
      data = nurses,
      family = binomial(link = "logit"))
```

02:00

An applied example

The data

Twitter!

- Real data, collected Wednesday, but anonymized
- You can see the code I used to get the data, but you'll pull different data if you run it
- 18,000 tweets including the hashtag #blm
- Sentence–level text coded for sentiment using the **{sentimentr}** package, then averaged for the entire tweet

Data prep

- Tweets of exactly 0 (neutral) sentiment removed
- Collapsed to positive/negative sentiment
- A few other features pulled out too (e.g., is trump mentioned in the person's bio)

Read in the data

It's a little different because there's still a list column of hashtags. Use code like the following:

```
library(tidyverse)
blm <- read_rds(here::here("data", "blm_sentiment.Rds"))
blm
```

```
## # A tibble: 14,339 x 21
##    user_id status_id trump_in_description followers_count friends_count
##      <dbl>     <dbl> <lgl>                          <int>         <int>
##  1    1691     14339 FALSE                            964           859
##  2    7740     14338 FALSE                            135           389
##  3     313     14337 FALSE                            261            31
##  4    5740     14336 FALSE                           4032          3872
##  5    5740      2927 FALSE                           4032          3872
##  6    5740      9379 FALSE                           4032          3872
##  7    5740      2457 FALSE                           4032          3872
##  8    5740      7854 FALSE                           4032          3872
##  9    5740      1550 FALSE                           4032          3872
## 10    5740     11789 FALSE                           4032          3872
## # … with 14,329 more rows, and 11 more variables: tweet_created_at <dttm
## #   n_mentions <int>, hashtags <list>, favorite_count <int>, retweet_cou
```

Getting more info

This is a data frame like any other with one exception – the `hashtags` column is a list.

See all hashtags

```
blm %>%
  unnest(hashtags) %>%
  count(hashtags, sort = TRUE) # %>%
```

```
## # A tibble: 12,011 x 2
##    hashtags             n
##    <chr>            <int>
##  1 BLM              12209
##  2 blm               2231
##  3 BlackLivesMatter  2153
##  4 GeorgeFloyd       1473
##  5 SashaJohnson       559
##  6 racism             416
##  7 blacklivesmatter   410
##  8 LGBTQ              360
##  9 FBR                291
## 10 Resist             289
```

List column

We can `unnest()` to see all of them, but we can't use that in modeling

Pull more features

Let's get the number of hashtags in the tweet

```
blm <- blm %>%
  rowwise() %>%
  mutate(n_hashtags = length(hashtags)) %>%
  ungroup()

blm %>%
  select(user_id, n_hashtags)
```

```
## # A tibble: 14,339 x 2
##    user_id n_hashtags
##      <dbl>      <int>
##  1    1691          1
##  2    7740          1
```

Antifa hashtag?

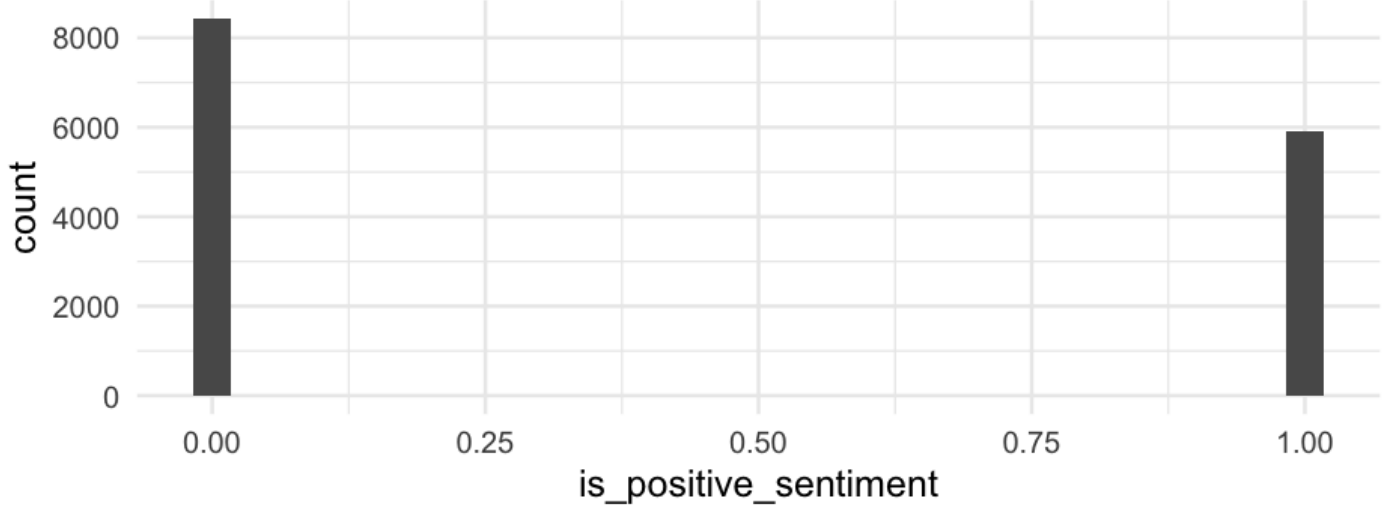
```
blm <- blm %>%
  rowwise() %>%
  mutate(has_antifa_hashtag = any(
    grepl("antifa", tolower(hashtags))
    )
  ) %>%
  ungroup()

blm %>%
  count(has_antifa_hashtag)
```

```
## # A tibble: 2 x 2
##   has_antifa_hashtag     n
##   <lgl>              <int>
## 1 FALSE              13786
## 2 TRUE                 553
```

Data exploration

Can we use some of these features to predict whether the sentiment of the tweet is positive? Let's explore the data some. First, look at the outcome:

```
ggplot(blm, aes(is_positive_sentiment)) +
  geom_histogram()
```



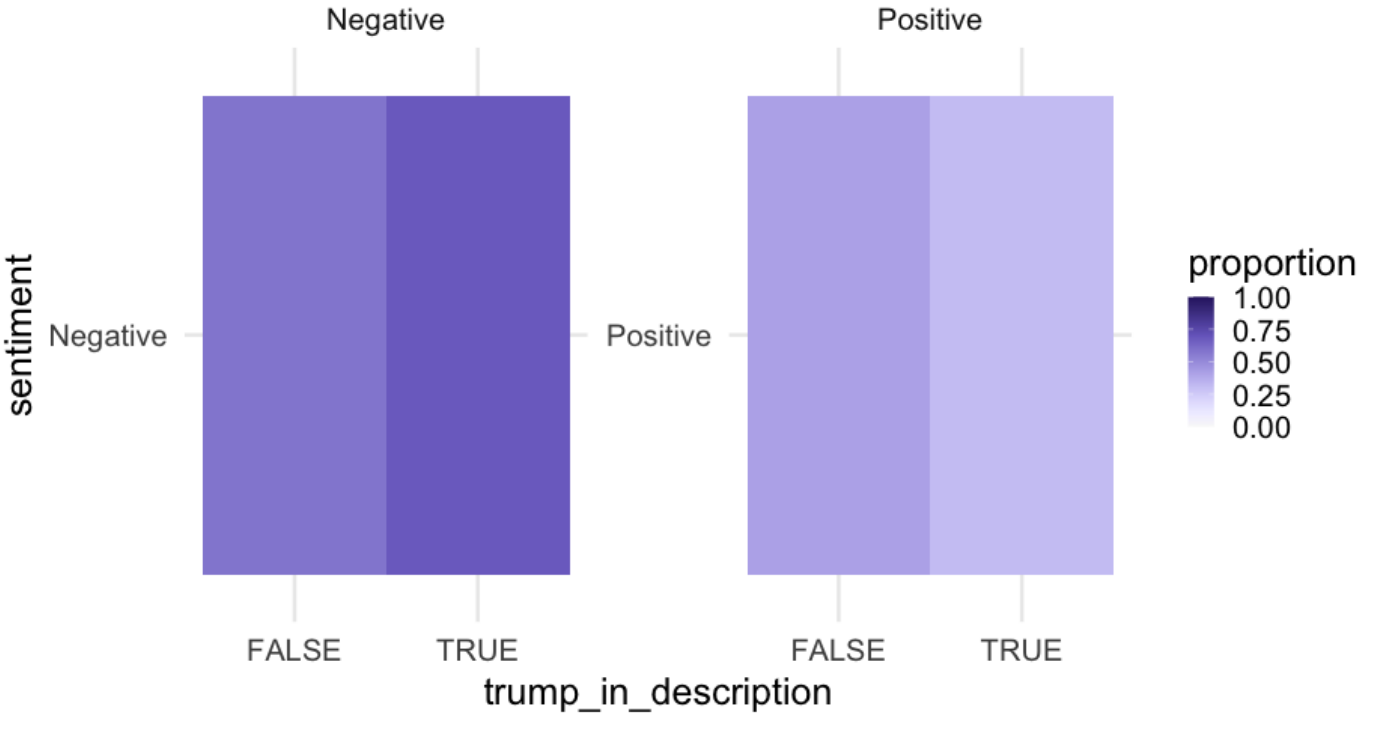
What about `trump_in_description`?

```
trump_proportions <- blm %>%
  mutate(sentiment = ifelse(
    is_positive_sentiment > 0, "Positive", "Negative"
    )
  ) %>%
  count(trump_in_description, sentiment) %>%
  group_by(trump_in_description) %>%
  mutate(proportion = n/sum(n))
trump_proportions
```

```
## # A tibble: 4 x 4
## # Groups:   trump_in_description [2]
##   trump_in_description sentiment     n proportion
##   <lgl>                <chr>     <int>      <dbl>
## 1 FALSE                Negative   8130  0.5848921
## 2 FALSE                Positive   5770  0.4151079
## 3 TRUE                 Negative    301  0.6856492
## 4 TRUE                 Positive    138  0.3143508
```

Visualize it

```
library(colorspace)
ggplot(trump_proportions, aes(trump_in_description, sentiment)) +
  geom_tile(aes(fill = proportion)) +
  scale_fill_continuous_sequential(palette = "Purples 3", limits
  facet_wrap(~sentiment, scales = "free_y")
```



Quick skim

Particularly when you're working with data that you're not **super** familiar with, `skimr::skim()` can be really helpful. Try it now!

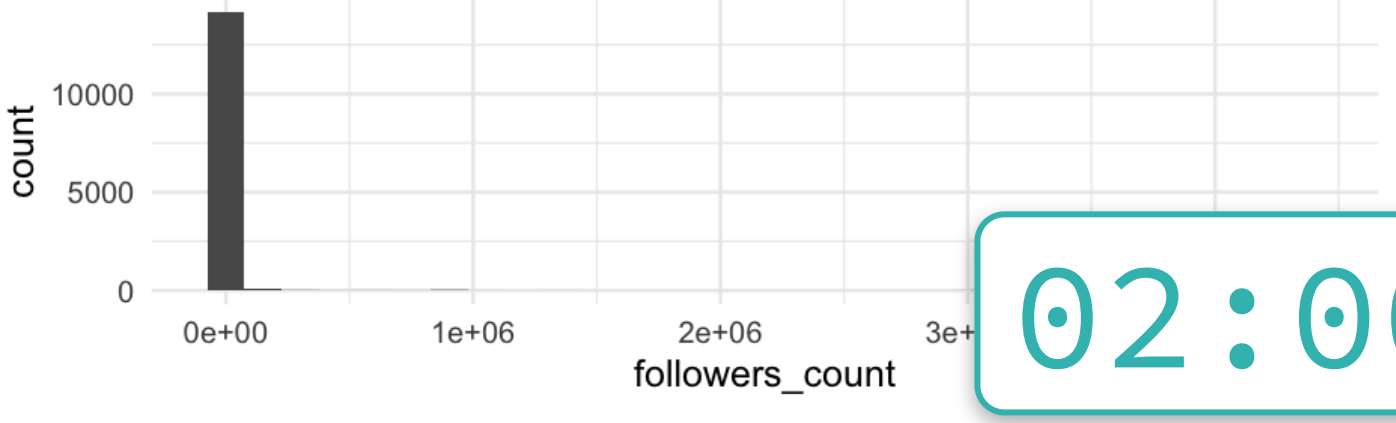
```
# install.packages("skimr")
skimr::skim(blm)
```

Distributions

Notice from `skimr::skim()` that some of the distributions are *highly* skewed, e.g., `followers_count`.

Can transformations help? Give it a try and see what you think

```
ggplot(blm, aes(followers_count)) +
  geom_histogram()
```

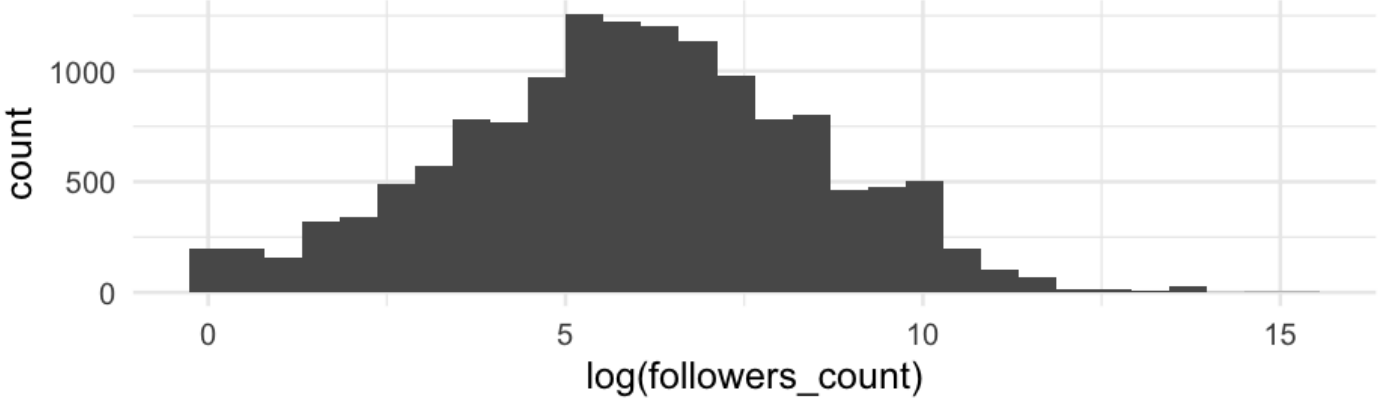


02:00

Log transformation

Note – it's not strictly neccessary for these to be normally distributed, but it can often help with estimation (while also potentially hurting interpretation, unless you're careful)

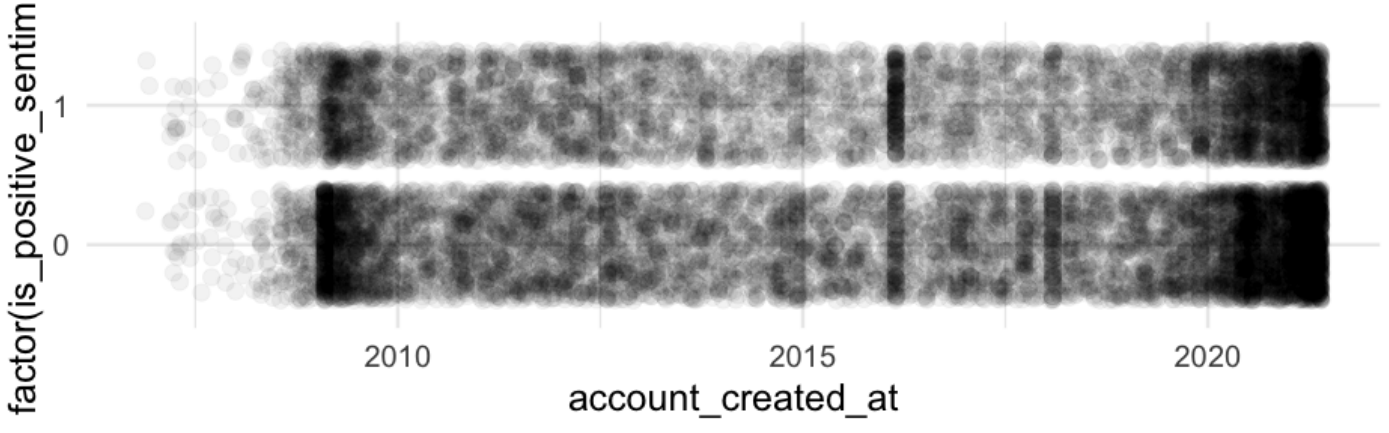
```
ggplot(blm, aes(log(followers_count))) +
  geom_histogram()
```



Account creation

In reality I would probably explore my data for a bit longer, unless I already knew a lot about out. For now, let's just do one more, looking at the relation between when their account was created, and whether the sentiment was positive.

```
ggplot(blm, aes(account_created_at, factor(is_positive_sentiment)
  geom_jitter(width = 0,
              alpha = 0.05)
```



Recent accounts only

Maybe a little bit of evidence...

```
blm %>%
  filter(account_created_at > lubridate::mdy("01/01/2020")) %>%
  ggplot(aes(account_created_at, factor(is_positive_sentiment)))
  geom_jitter(width = 0, alpha = 0.2)
```

Last bit

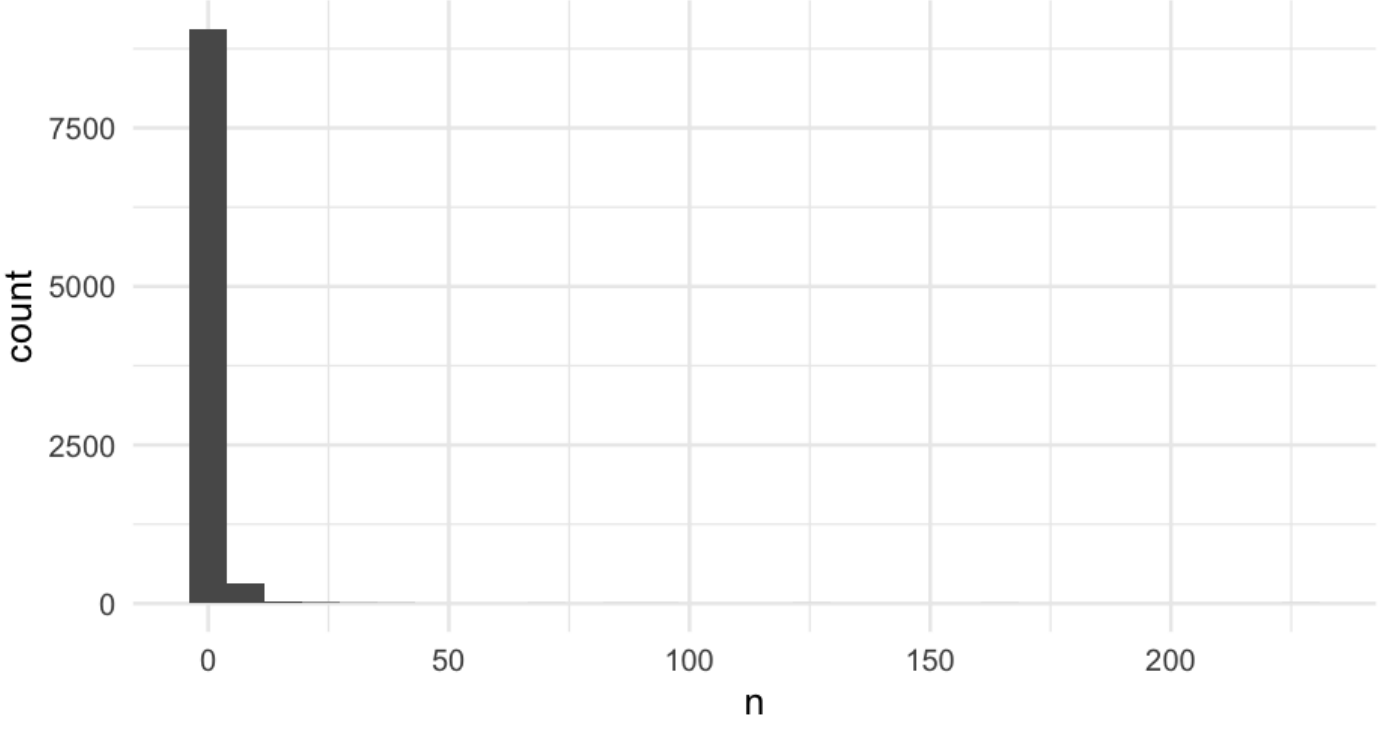
Sample size issues

The number of **tweets per person** varies a lot.

- When $n = 1$ it's not theoretically a problem, although I've had issues with estimation in the past.
- You might consider including the number of tweets a person as a predictor.
    - Could be an indicator they are a bot or a journalist.

See here for more information

```
blm %>%
  count(user_id) %>%
  ggplot(aes(n)) +
  geom_histogram()
```



Modeling

Person–variance

We have lots of tweets from lots of people – maybe we start by modeling the baseline variability in sentiment across people?

You try first – go ahead and just use **{lme4}** and then we'll replicate it with **{brms}**

02:00

Maximum likelihood version

```
library(lme4)
m0_ml <- glmer(is_positive_sentiment ~ 1 + (1|user_id),
               data = blm,
               family = binomial(link = "logit"))
arm::display(m0_ml)
```

```
## glmer(formula = is_positive_sentiment ~ 1 + (1 | user_id), data = blm,
##     family = binomial(link = "logit"))
## coef.est  coef.se
##    -0.40     0.02
##
## Error terms:
##  Groups   Name        Std.Dev.
##  user_id  (Intercept) 0.95
##  Residual             1.00
## ---
## number of obs: 14339, groups: user_id, 9454
## AIC = 18757.7, DIC = 10514.1
## deviance = 14633.9
```

Interpretation

The baseline log–odds of a positive tweet was –0.40. The `brms::inv_logit_scaled()` function will translate it to probability.

```
brms::inv_logit_scaled(fixef(m0_ml))
```

```
## (Intercept)
##   0.4004497
```

So about a 40% chance.

Variability

The log–odds varied between people with a standard deviation of 0.95.

The probability of a person one standard deviation above and below the average posting a positive tweet were estimated at:

```
# Probability for an individual 1 SD below
brms::inv_logit_scaled(-0.40 - 0.95)
```

```
## [1] 0.2058704
```

```
# Probability for an individual 1 SD above
brms::inv_logit_scaled(-0.40 + 0.95)
```

```
## [1] 0.6341356
```

Plot the variability

First pull the random effect estimates (deviations from the fixed effect)

```
library(broom.mixed)
tidy_m0_ml <- tidy(m0_ml, "ran_vals", conf.int = TRUE) %>%
  mutate(level = fct_reorder(level, estimate))
```

Next create the plot

```
ggplot(tidy_m0_ml, aes(estimate, level)) +
  geom_linerange(aes(xmin = conf.low, xmax = conf.high),
                 alpha = 0.01) +
  geom_point(color = "#1DA1F2") +

  # get rid of some plot elements
  theme(axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        panel.grid.major.y = element_blank(),
        panel.grid.minor = element_blank())
```

Plot the variability

Fitting with {brms}

We would probably actually just pick a framework and go with that, **but**, in my experience, multilevel binomial models often have a hard time with convergence. Bayes can help with that.

Let's re–fit with **{brms}**

Fit using Bayes

You try first. Fit the same model using Bayes. Go ahead and assume flat priors.

```
library(brms)
m0_b <- brm(is_positive_sentiment ~ 1 + (1|user_id),
            data = blm,
            family = bernoulli(link = "logit"),
            cores = 4,
            backend = "cmdstanr")
```

```
##
-
\
|
/
-
\
|
```

04:00

Summary

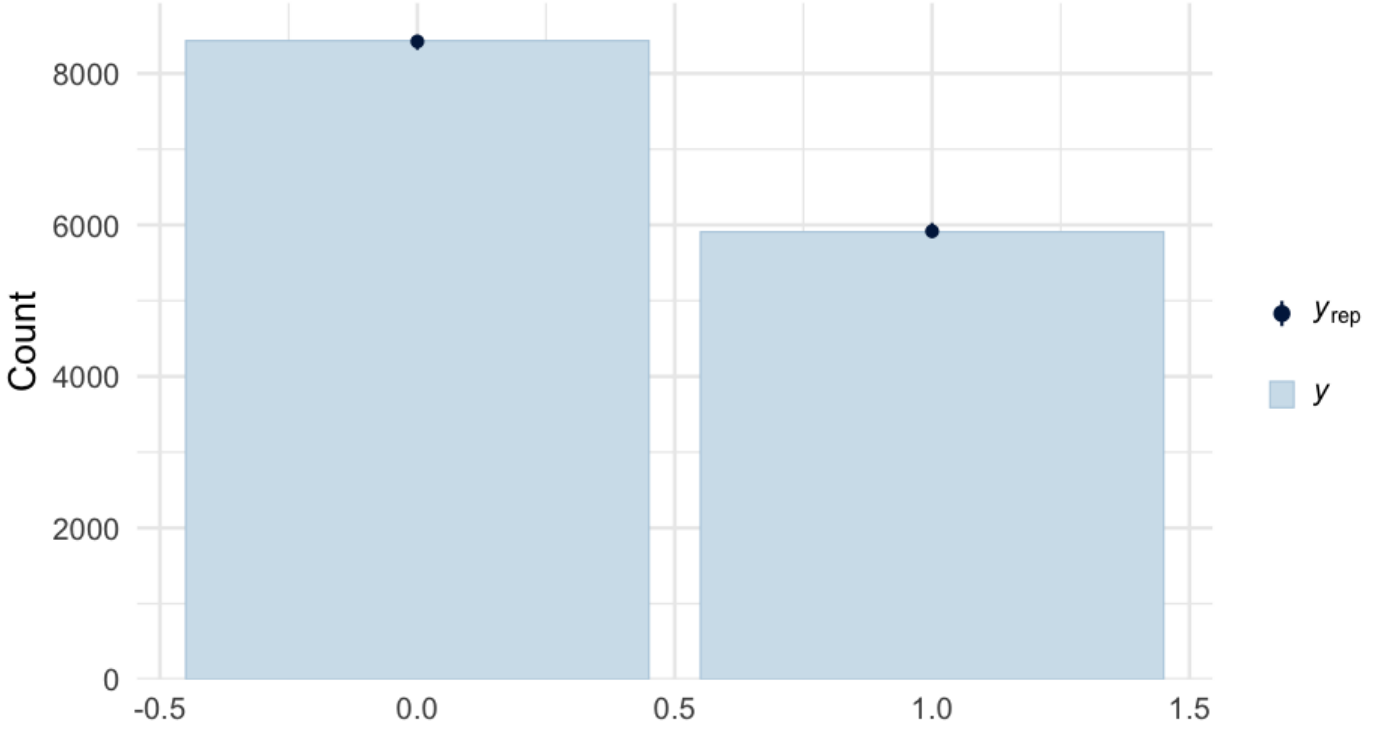
Notice the variance is fairly different here

```
summary(m0_b)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: is_positive_sentiment ~ 1 + (1 | user_id)
##    Data: blm (Number of observations: 14339)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Group-Level Effects:
## ~user_id (Number of levels: 9454)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ES
## sd(Intercept)     1.51      0.07     1.38     1.65 1.01      528     122
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -0.46      0.03    -0.51    -0.40 1.00     2643     2906
##
## Samples were drawn using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potenti
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

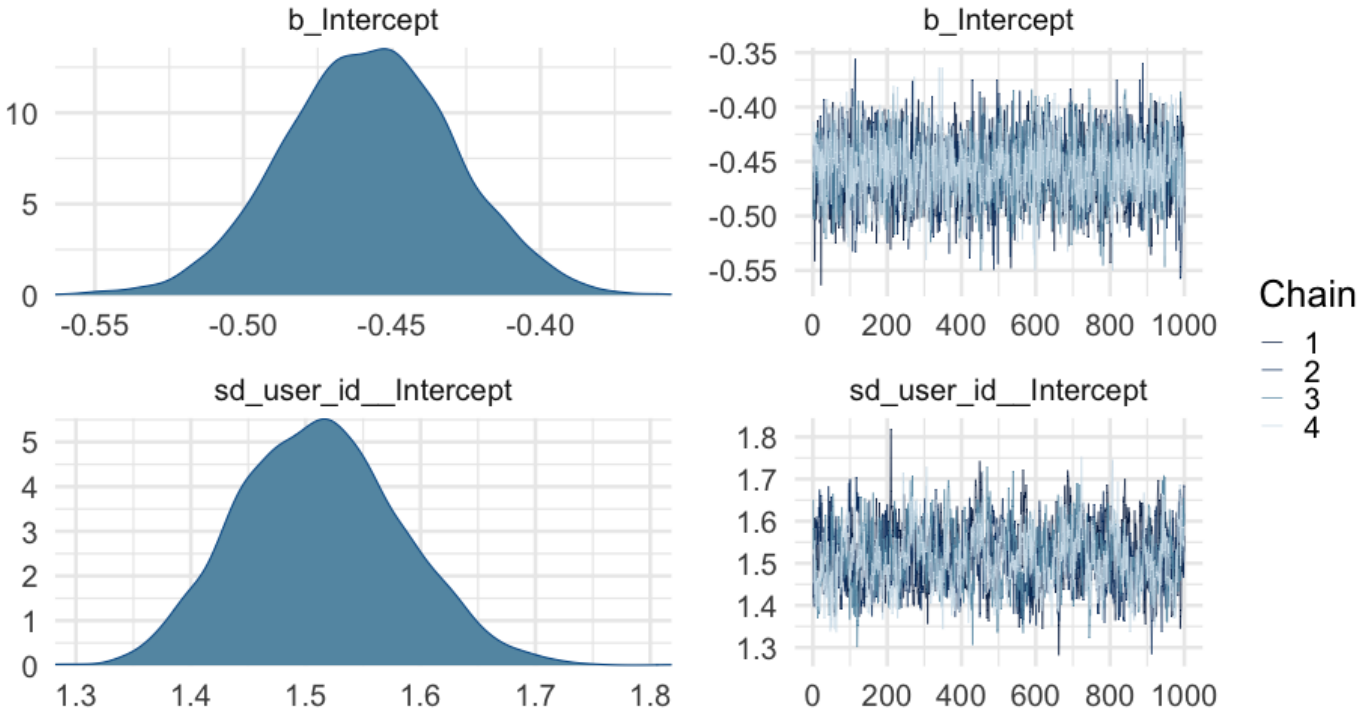
Posterior predictive

```
pp_check(m0_b, type = "bars")
```



Convergence checks

```
plot(m0_b)
```



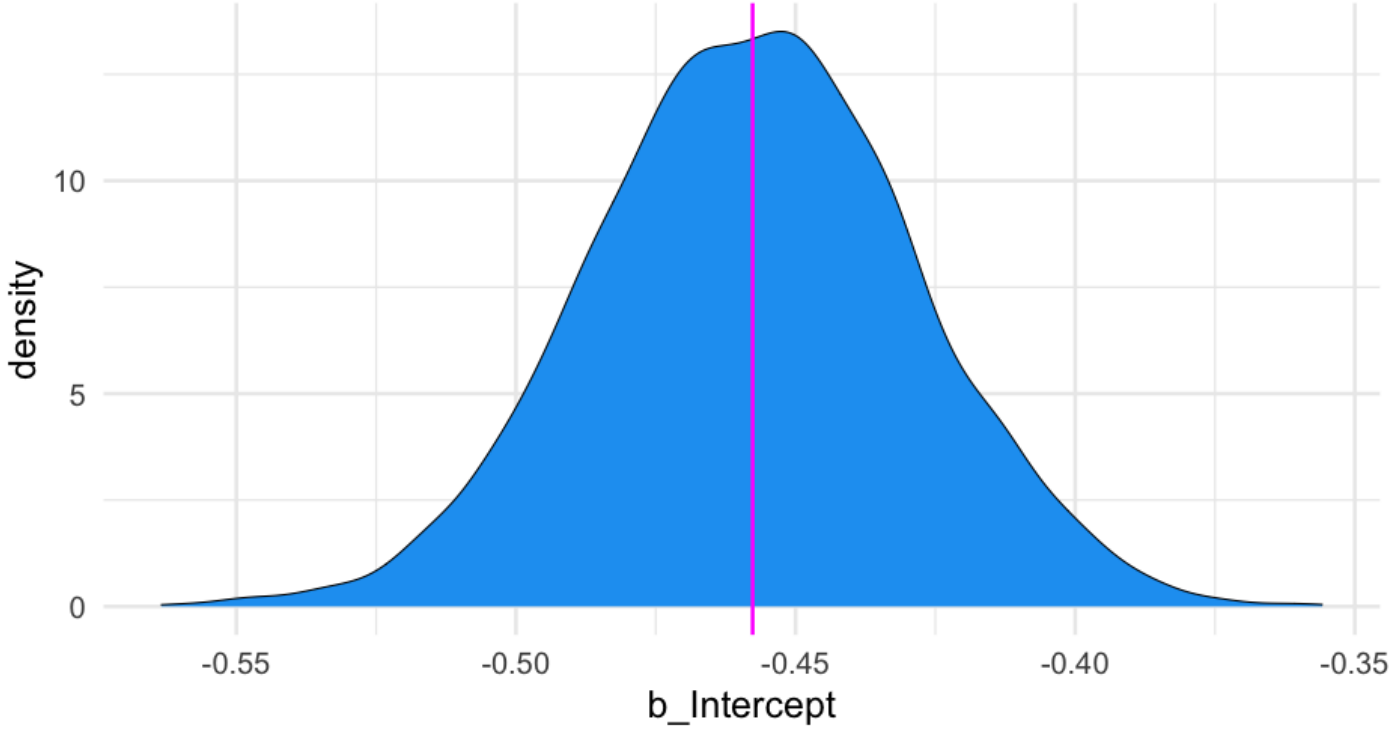
Posteriors

```
library(insight)
m0_posterior <- get_parameters(m0_b)
head(m0_posterior)
```

```
##   b_Intercept
## 1   -0.500646
## 2   -0.454907
## 3   -0.463268
## 4   -0.497465
## 5   -0.541269
## 6   -0.472217
```

Plot density

```
ggplot(m0_posterior, aes(b_Intercept)) +
  geom_density(fill = "#1DA1F2") +
  geom_vline(aes(xintercept = mean(b_Intercept)),
             color = "magenta",
             size = 1.2)
```



Using the density

What's the likelihood that the intercept (baseline log–odds) is *less than –0.5* (i.e., average probability of a positive tweet less than 0.38)?

```
sum(m0_posterior$b_Intercept < -0.5) / nrow(m0_posterior)
```

```
## [1] 0.06925
```

About a 7% chance

Plot person–estimates

We have to go to **{tidybayes}** for this

- General purpose tool to pull lots of different things from our model and plot them
- For now, we'll do the plotting ourselves
- Let's start by looking at what's actually in the model

In this case `r_*` implies "random". These are the deviations from the average.

```
library(tidybayes)
get_variables(m0_b)
```

```
##    [1] "b_Intercept"                "sd_user_id__Intercept"    "Intercept"
##    [6] "r_user_id[3,Intercept]"     "r_user_id[4,Intercept]"   "r_user_id[
##   [11] "r_user_id[8,Intercept]"     "r_user_id[9,Intercept]"   "r_user_id[
```

Pull random vars

- The random effect name is `r_user_id`
- We use brackets to assign new names

```
m0_id_re <- gather_draws(m0_b, r_user_id[id, term])
m0_id_re
```

```
## # A tibble: 37,816,000 x 7
## # Groups:   id, term, .variable [9,454]
##       id term      .chain .iteration .draw .variable     .value
##    <int> <chr>      <int>      <int> <int> <chr>          <dbl>
##  1     1 Intercept      1          1     1 r_user_id -0.0826422
##  2     1 Intercept      1          2     2 r_user_id -0.567424
##  3     1 Intercept      1          3     3 r_user_id -0.546465
##  4     1 Intercept      1          4     4 r_user_id -1.88725
##  5     1 Intercept      1          5     5 r_user_id  1.12419
##  6     1 Intercept      1          6     6 r_user_id -2.31118
##  7     1 Intercept      1          7     7 r_user_id -0.249097
##  8     1 Intercept      1          8     8 r_user_id -3.61696
##  9     1 Intercept      1          9     9 r_user_id -1.6264
## 10     1 Intercept      1         10    10 r_user_id  0.158484
## # … with 37,815,990 more rows
```

Compute credible intervals

I recognize this part is complex, but you'll have the code for reference

```
id_qtiles <- m0_id_re %>%
  group_by(id) %>%
  summarize(
    probs = c("median", "lower", "upper"),
    qtiles = quantile(.value,probs = c(0.5, 0.025, 0.975))
  ) %>%
  ungroup()
id_qtiles
```

```
## # A tibble: 28,362 x 3
##       id probs      qtiles
##    <int> <chr>       <dbl>
##  1     1 median -0.6739
##  2     1 lower  -3.31484
##  3     1 upper   1.746887
##  4     2 median  0.9492155
##  5     2 lower  -1.605349
##  6     2 upper   3.453275
##  7     3 median -0.648632
##  8     3 lower  -3.341358
```

Move it wider

```
id_qtiles <- id_qtiles %>%
  pivot_wider(names_from = "probs",
              values_from = "qtiles") %>%
  mutate(id = fct_reorder(factor(id), median))

id_qtiles
```

```
## # A tibble: 9,454 x 4
##    id        median     lower    upper
##    <fct>      <dbl>     <dbl>    <dbl>
##  1 1     -0.6739    -3.31484  1.746887
##  2 2      0.9492155 -1.605349 3.453275
##  3 3     -0.648632  -3.341358 1.803793
##  4 4      0.9214355 -1.517492 3.542017
##  5 5     -0.626921  -3.264422 1.725803
##  6 6      0.930557  -1.599318 3.539588
##  7 7     -0.6482415 -3.349034 1.817699
##  8 8     -0.6443055 -3.212276 1.697759
##  9 9     -0.6500125 -3.371763 1.892287
## 10 10    -1.029665  -3.648958 1.221002
## # … with 9,444 more rows
```

Plot

```
ggplot(id_qtiles, aes(median, id)) +
  geom_linerange(aes(xmin = lower, xmax = upper),
                 alpha = 0.01) +
  geom_point(color = "#1DA1F2") +
  theme(axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        panel.grid.major.y = element_blank(),
        panel.grid.minor = element_blank())
```

```
ggplot(id_qtiles, aes(median, id)) +
  geom_linerange(aes(xmin = lower, xmax = upper),
                 alpha = 0.01) +
  geom_point(color = "#1DA1F2") +
  theme(axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        panel.grid.major.y = element_blank(),
        panel.grid.minor = element_blank())
```

Extend our model

Let's add the following predictors to our model:

- `trump_in_description`
- `has_antifa_hashtag`
- `log(favorite_count + 1)`

You try first

Just try writing the code – not running the model.

01:00

```
m1_b <- brm(is_positive_sentiment ~ trump_in_description +
              has_antifa_hashtag + log(favorite_count + 1) +
              (1|user_id),
            data = blm,
            family = bernoulli(link = "logit"),
            cores = 4,
            backend = "cmdstanr")
```

```
##
-
\
|
/
-
\
|
/
-
\
|
```

Summary

```
summary(m1_b)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: is_positive_sentiment ~ trump_in_description + has_antifa_hasht
##    Data: blm (Number of observations: 14339)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Group-Level Effects:
## ~user_id (Number of levels: 9454)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ES
## sd(Intercept)     1.51      0.07     1.37     1.65 1.01      663     134
##
## Population-Level Effects:
##                          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_
## Intercept                   -0.47      0.03    -0.54    -0.40 1.00     2
## trump_in_descriptionTRUE    -0.36      0.19    -0.74     0.01 1.00     2
## has_antifa_hashtagTRUE      -0.35      0.13    -0.61    -0.10 1.00     3
## logfavorite_countP1          0.05      0.03    -0.00     0.10 1.00     3
##
## Samples were drawn using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potenti
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Posteriors

What's the likelihood that our posterior mean for the log of favorite counts is positive?

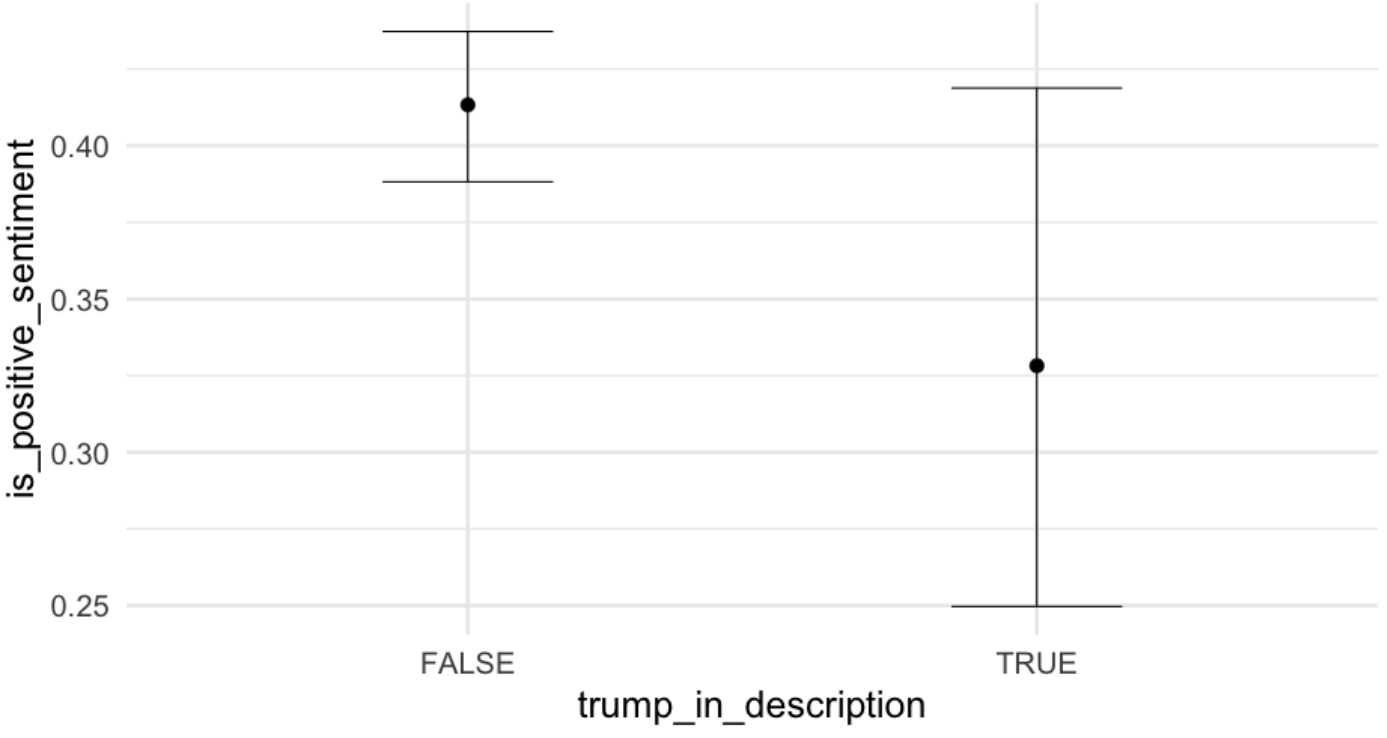
```
m1_posterior <- get_parameters(m1_b)
sum(m1_posterior$b_logfavorite_countP1 > 0) / nrow(m1_posterior)
```

```
## [1] 0.97175
```

97% probability! But note – this would probably just miss "significance", with a frequentist approach because of the use of two–tailed tests.

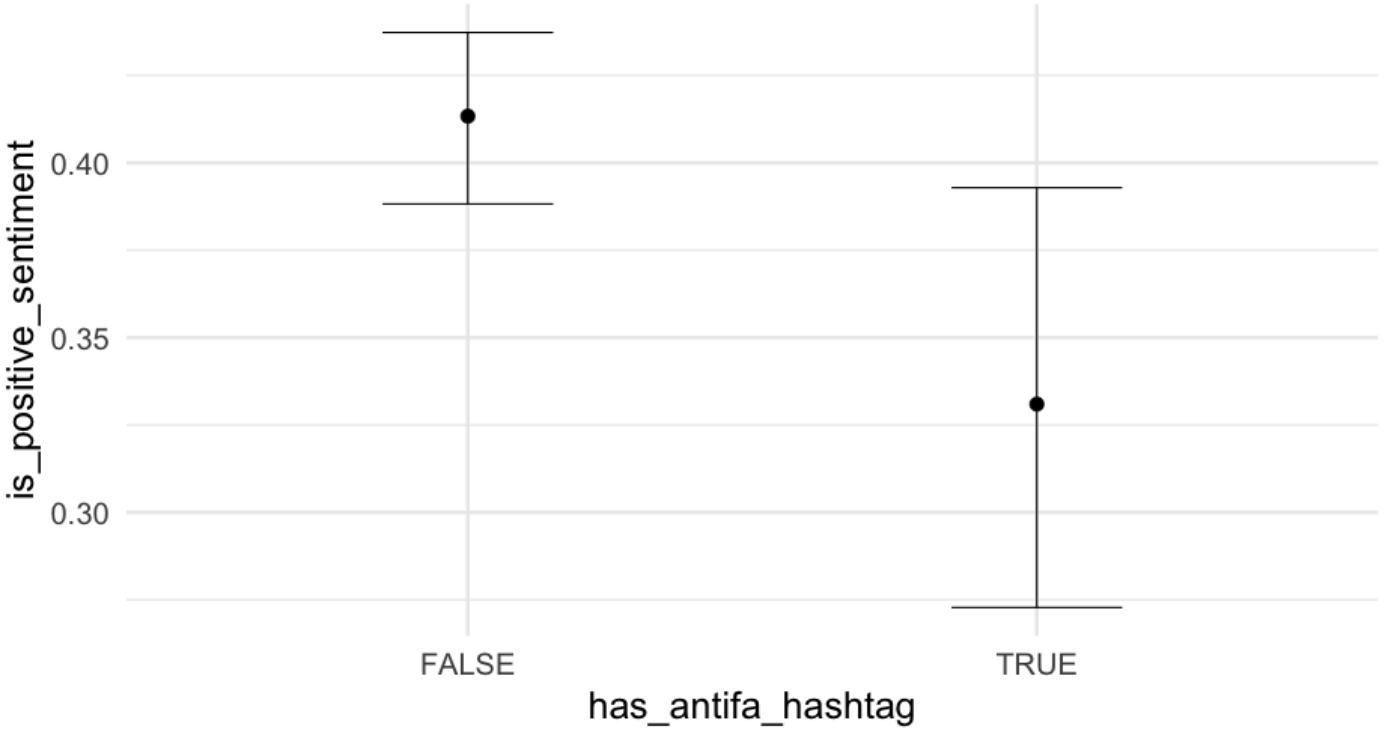
Marginal plots

```
conditional_effects(m1_b, "trump_in_description")
```



Marginal plots

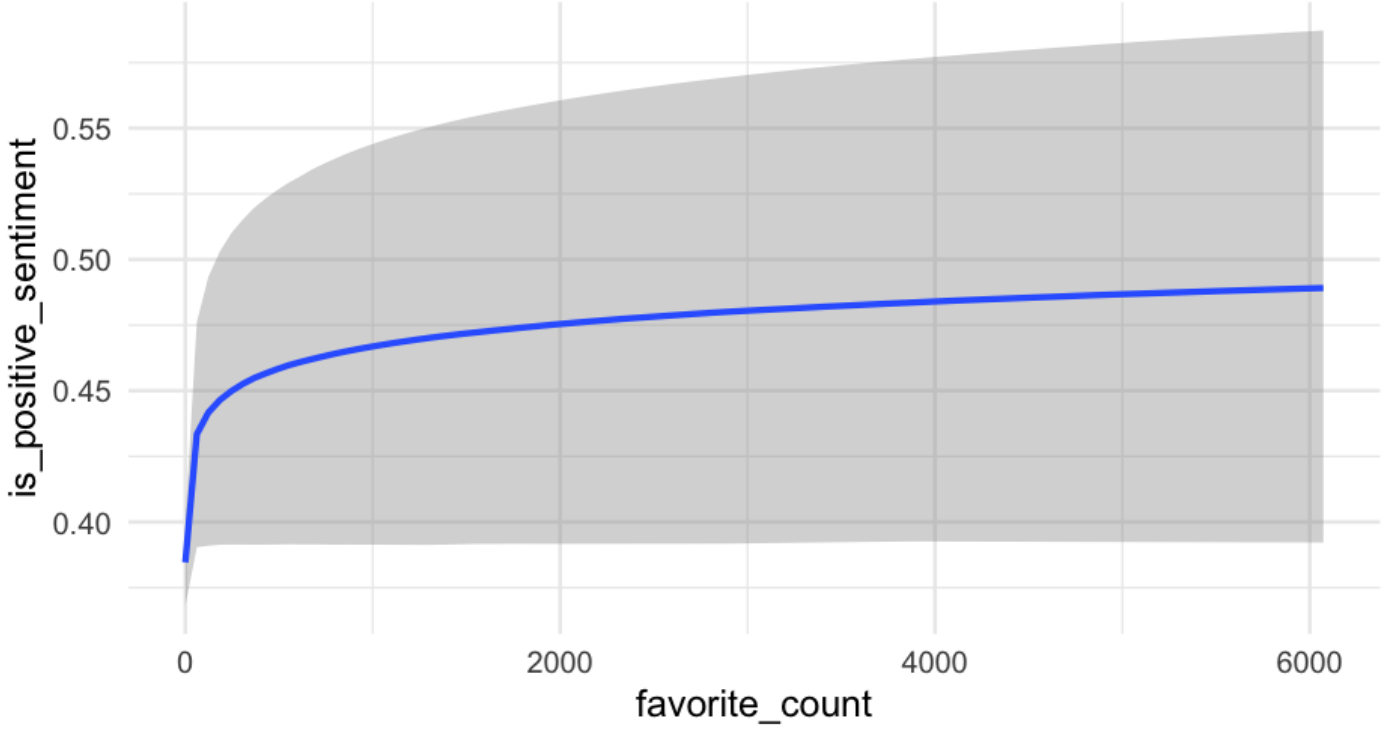
```
conditional_effects(m1_b, "has_antifa_hashtag")
```



Marginal plots

Notice this is on the raw scale, not the log scale

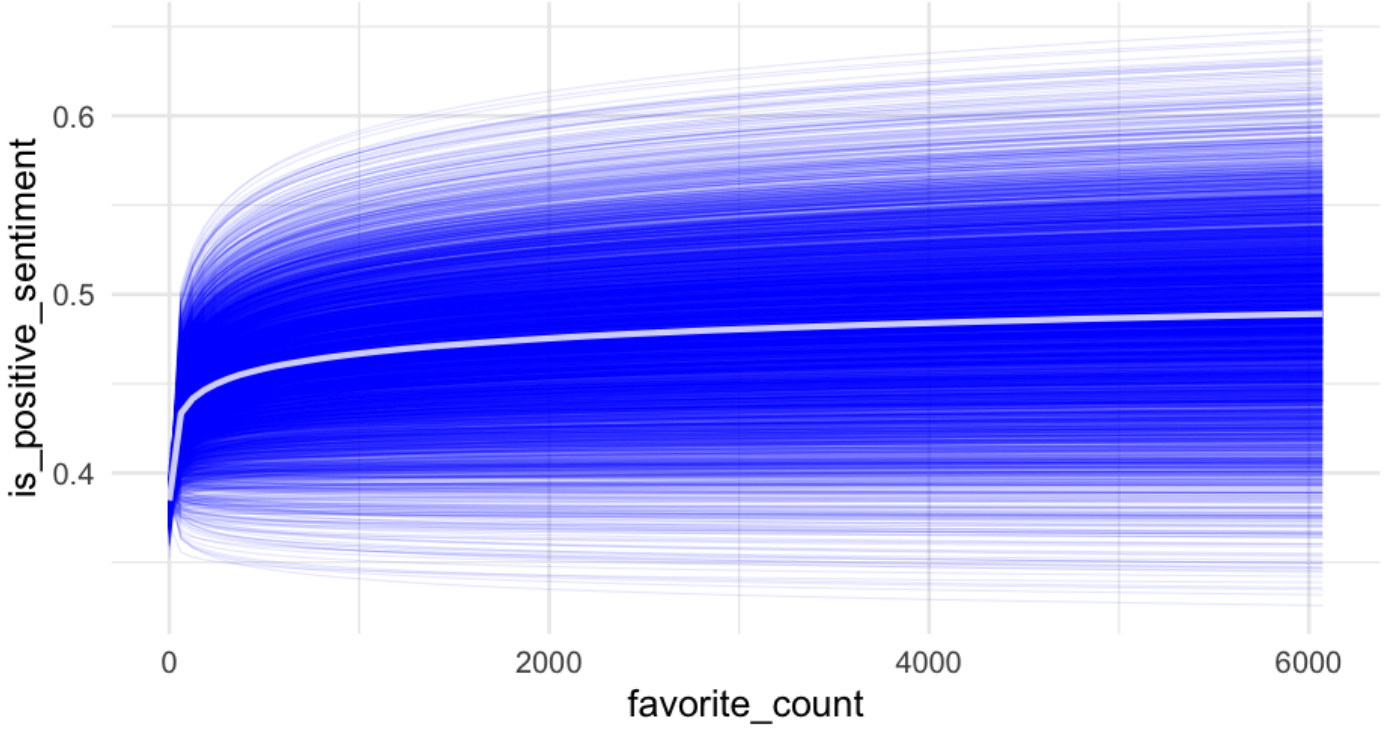
```
conditional_effects(m1_b, "favorite_count")
```



Marginal plots: Spaghetti

Notice this is on the raw scale, not the log scale

```
conditional_effects(m1_b, "favorite_count", spaghetti = TRUE)
```



Warnings

As I was playing around with different models, I sometimes ran into warnings.

These were easy to solve in this case by just log–transforming the predictor variables that were highly skewed.

For general guidance, see here.

Break

05:00

A second example

There's more we could do here, but this example (like lots of real data) is sort of difficult for illustration. Let's try a different dataset

New data

Lung cancer data: Patients nested in doctors

```
hdp <- read_csv("https://stats.idre.ucla.edu/stat/data/hdp.csv")
  janitor::clean_names() %>%
  select(did, tumorsize, pain, lungcapacity, age, remission)
hdp
```

```
## # A tibble: 8,525 x 6
##      did tumorsize  pain lungcapacity      age remission
##    <dbl>     <dbl> <dbl>        <dbl>    <dbl>     <dbl>
##  1     1  67.98120     4    0.8010882 64.96824         0
##  2     1  64.70246     2    0.3264440 53.91714         0
##  3     1  51.56700     6    0.5650309 53.34730         0
##  4     1  86.43799     3    0.8484109 41.36804         0
##  5     1  53.40018     3    0.8864910 46.80042         0
##  6     1  51.65727     4    0.7010307 51.92936         0
##  7     1  78.91707     3    0.8908539 53.82926         0
##  8     1  69.83325     3    0.6608795 46.56223         0
##  9     1  62.85259     4    0.9088714 54.38936         0
## 10     1  71.77790     5    0.9593268 50.54465         0
## # … with 8,515 more rows
```

Predict remission

Build a model where age, lung capacity, and tumor size predict whether or not the patient was in remission.

- Build the model so you can evaluate whether or not the relation between the tumor size and likelihood of remission depends on age
- Allow the intercept to vary by the doctor ID.
- Fit the model using **brms**

05:00

Lung cancer remission model

```
lc <- brm(
  remission ~ age * tumorsize + lungcapacity + (1|did),
  data = hdp,
  family = bernoulli(link = "logit"),
  cores = 4,
  backend = "cmdstan"
)
```

```
##
-

\

|

/

-

\
```

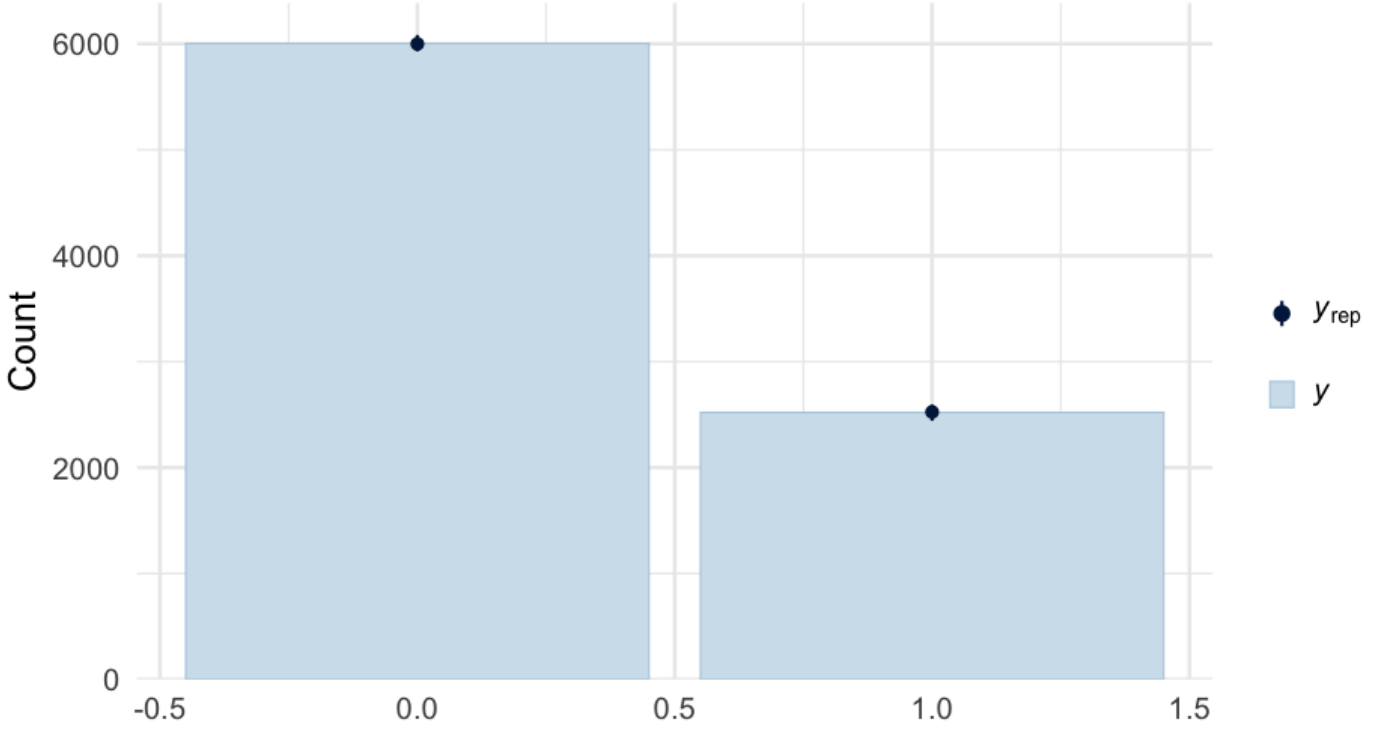
Model summary

```
summary(lc)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: remission ~ age * tumorsize + lungcapacity + (1 | did)
##    Data: hdp (Number of observations: 8525)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Group-Level Effects:
## ~did (Number of levels: 407)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ES
## sd(Intercept)     2.01      0.10     1.82     2.23 1.00      599     160
##
## Population-Level Effects:
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ES
## Intercept         1.66      1.51    -1.31     4.57 1.00     1925     278
## age              -0.05      0.03    -0.10     0.01 1.00     1919     253
## tumorsize        -0.01      0.02    -0.05     0.03 1.00     1931     283
## lungcapacity      0.07      0.19    -0.29     0.44 1.00     4872     306
## age:tumorsize    -0.00      0.00    -0.00     0.00 1.00     1905     281
##
## Samples were drawn using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potenti
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Posterior predictive check

```
pp_check(lc, type = "bars")
```

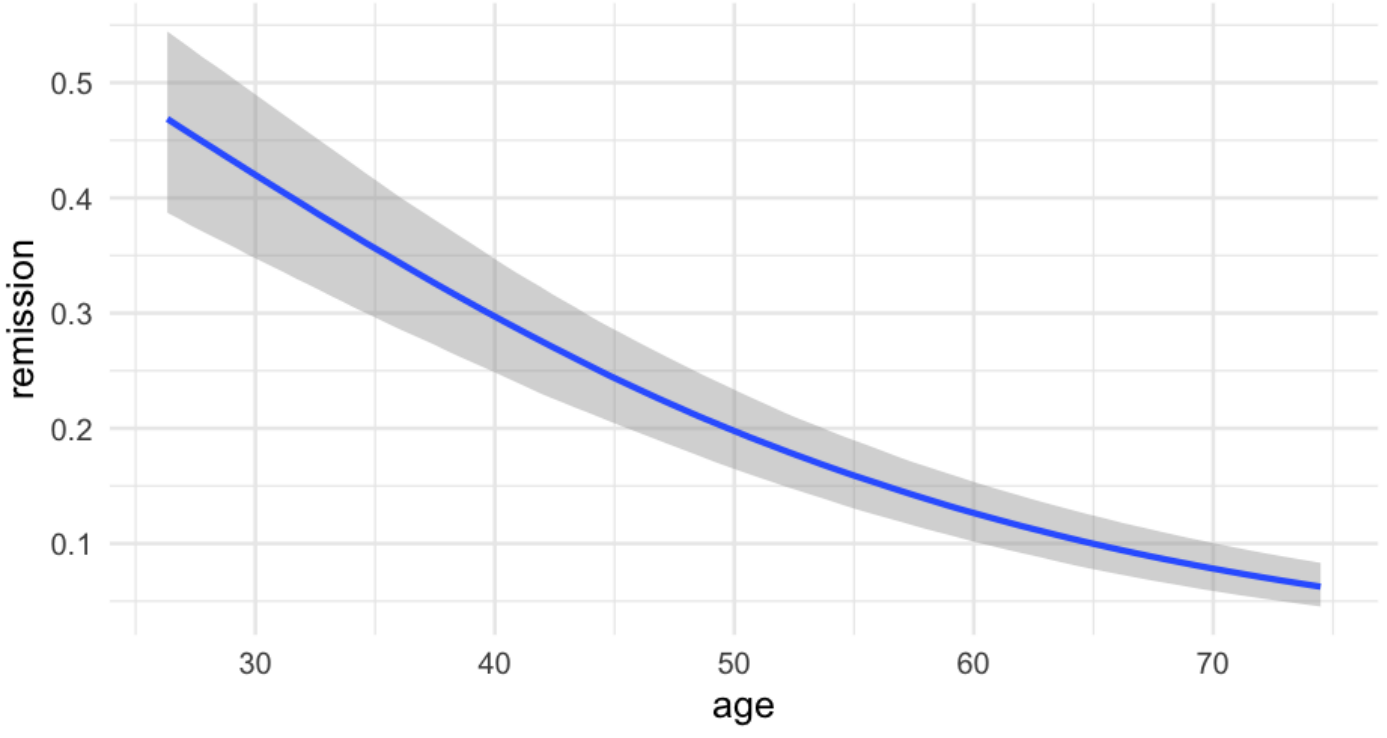


Chains

```
plot(lc)
```

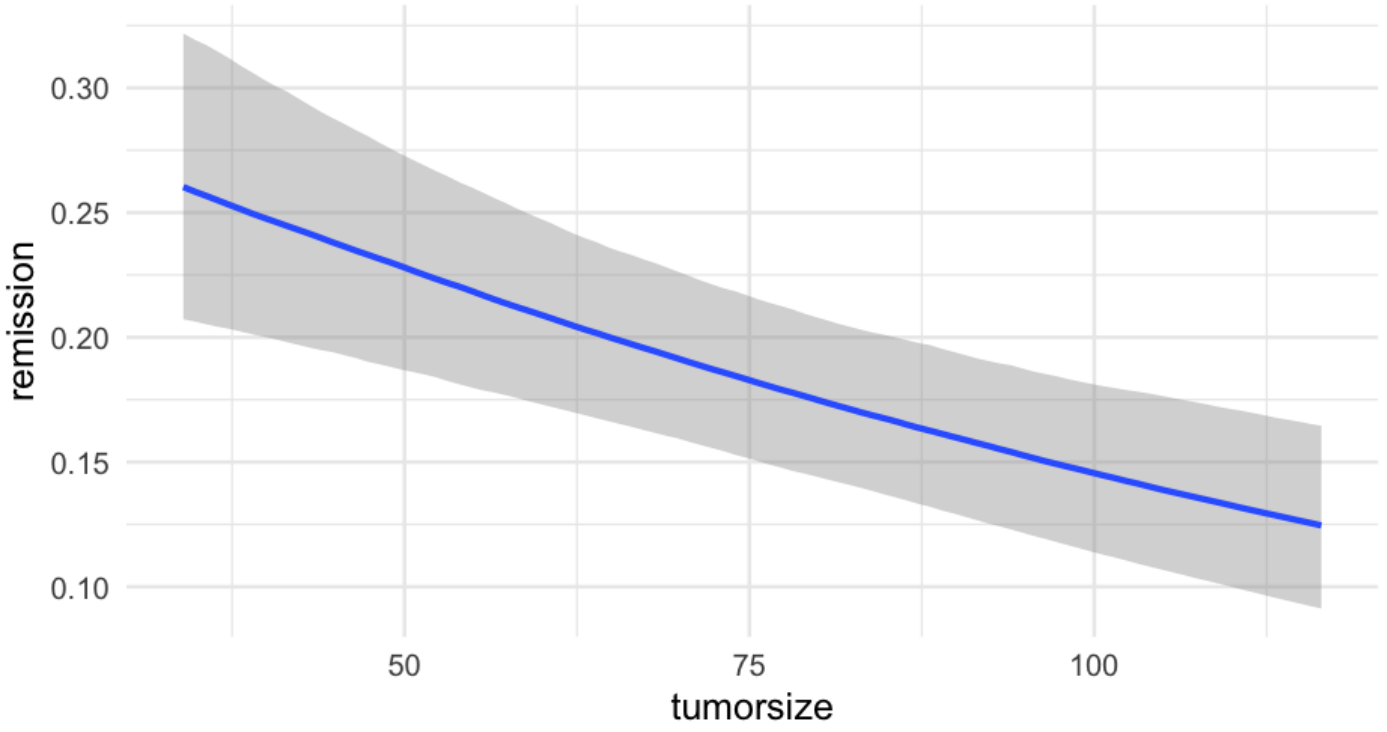
Marginal predictions: Age

```
conditional_effects(lc, "age")
```



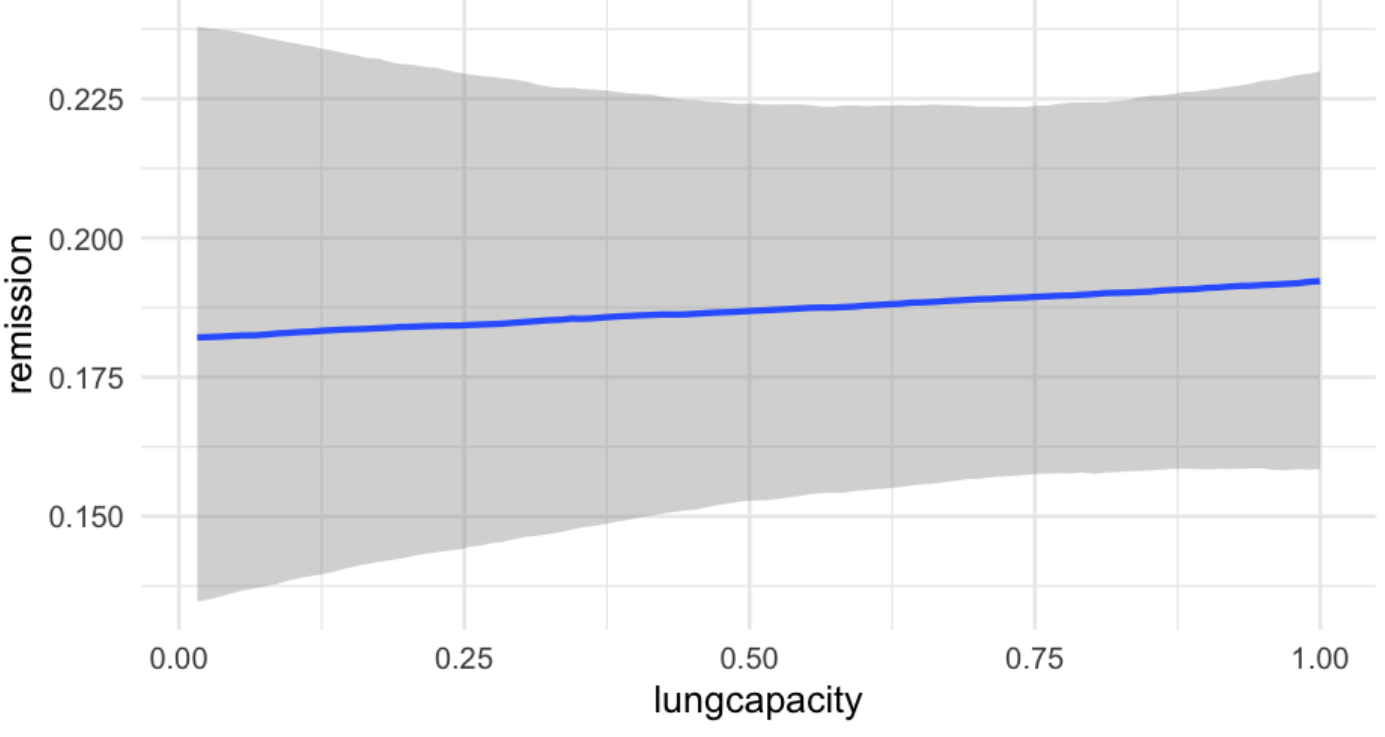
Marginal predictions: tumor size

```
conditional_effects(lc, "tumorsize")
```



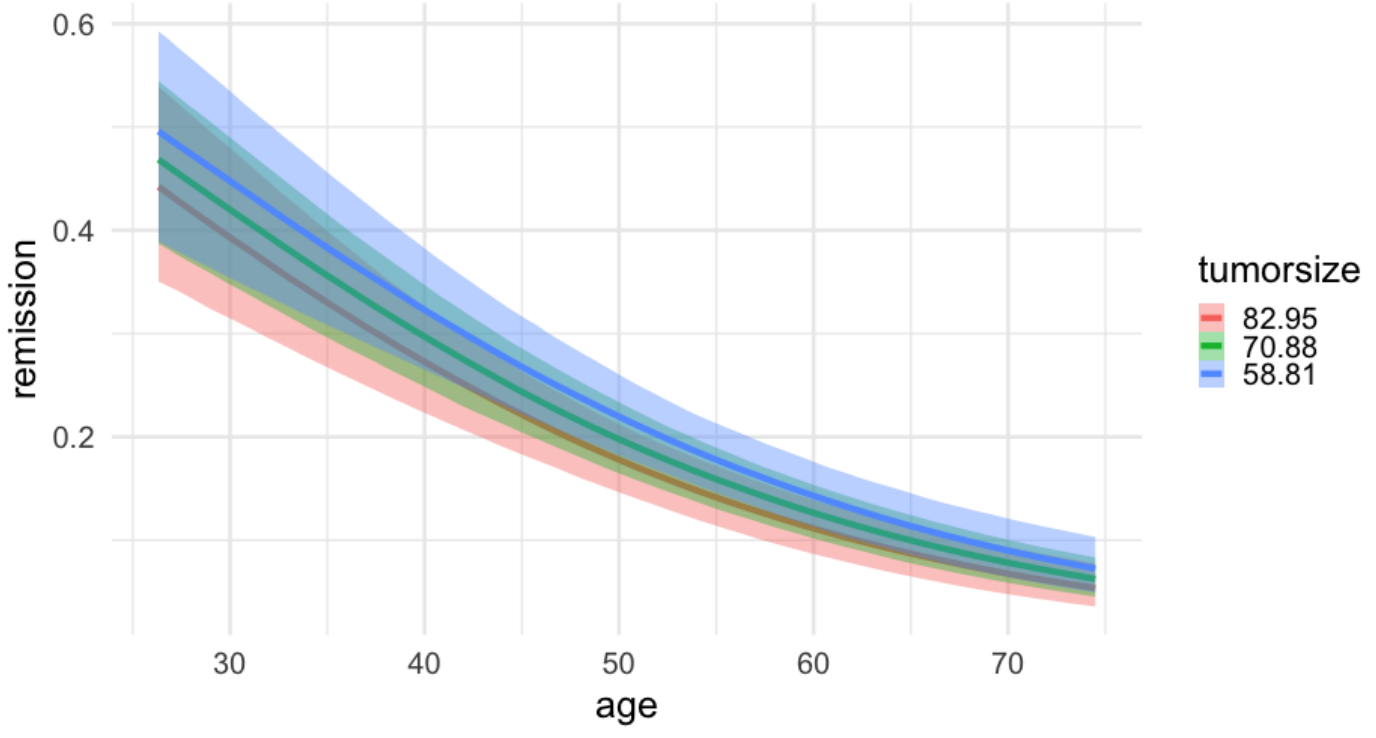
Marginal predictions: lung capacity

```
conditional_effects(lc, "lungcapacity")
```



Interaction

```
conditional_effects(lc, "age:tumorsize")
```



Make predictions

Check the relation for tumor size

Note – we're using **{tidybayes}** again

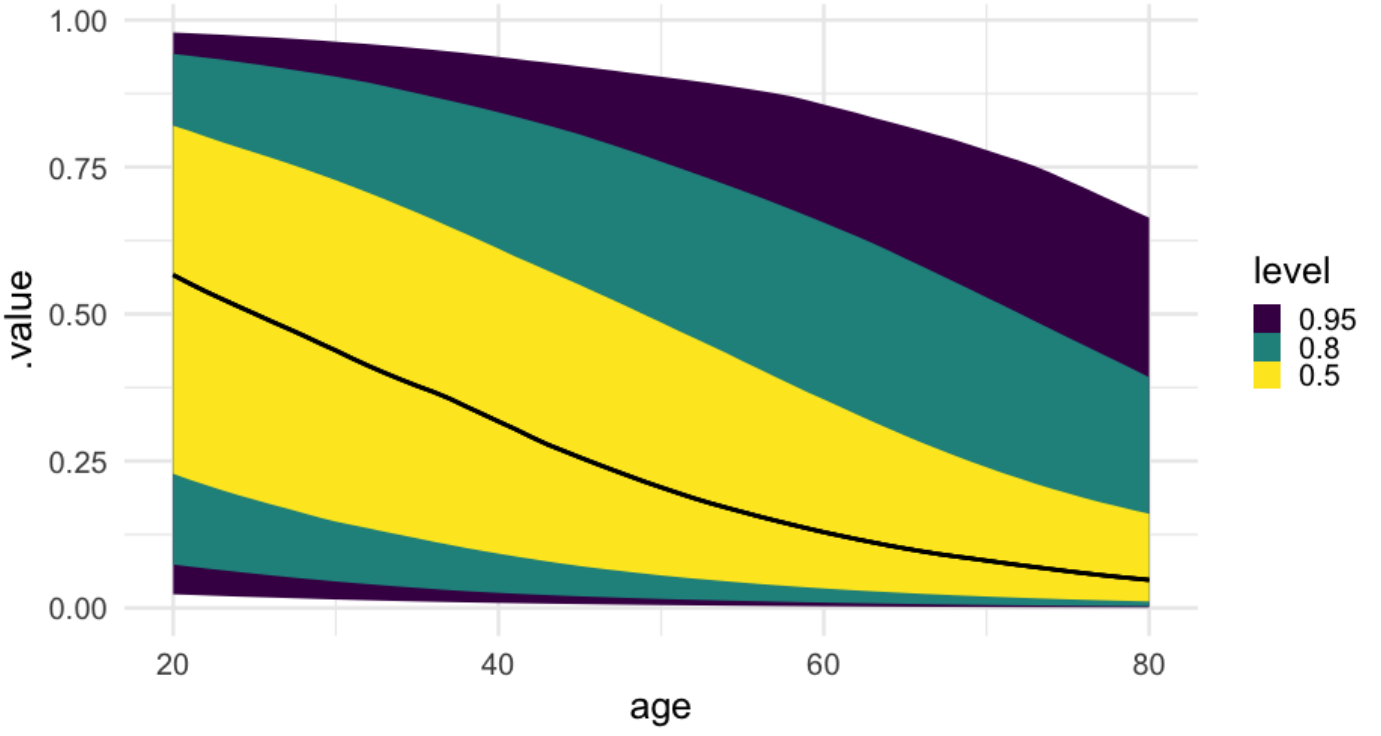
```
pred_tumor <- expand.grid(
    age = 20:80,
    lungcapacity = mean(hdp$lungcapacity),
    tumorsize = 30:120,
    did = -999
  ) %>%
  # tidybayes part
  add_fitted_draws(model = lc,
                   n = 100,
                   allow_new_levels = TRUE)
```

```
pred_tumor
```

```
## # A tibble: 555,100 x 9
## # Groups:   age, lungcapacity, tumorsize, did, .row [5,551]
##      age lungcapacity tumorsize   did  .row .chain .iteration .draw    .
##    <int>        <dbl>     <int> <dbl> <int>  <int>      <int> <int>
##  1    20    0.7740865        30  -999     1     NA         NA    22 0.88
##  2    20    0.7740865        30  -999     1     NA         NA   167 0.13
##  3    20    0.7740865        30  -999     1     NA         NA   296 0.45
##  4    20    0.7740865        30  -999     1     NA         NA   327 0.93
##  5    20    0.7740865        30  -999     1     NA         NA   371 0.31
##  6    20    0.7740865        30  -999     1     NA         NA   392 0.98
##  7    20    0.7740865        30  -999     1     NA         NA   446 0.99
##  8    20    0.7740865        30  -999     1     NA         NA   461 0.35
##  9    20    0.7740865        30  -999     1     NA         NA   555 0.87
## 10    20    0.7740865        30  -999     1     NA         NA   559 0.74
## # … with 555,090 more rows
```

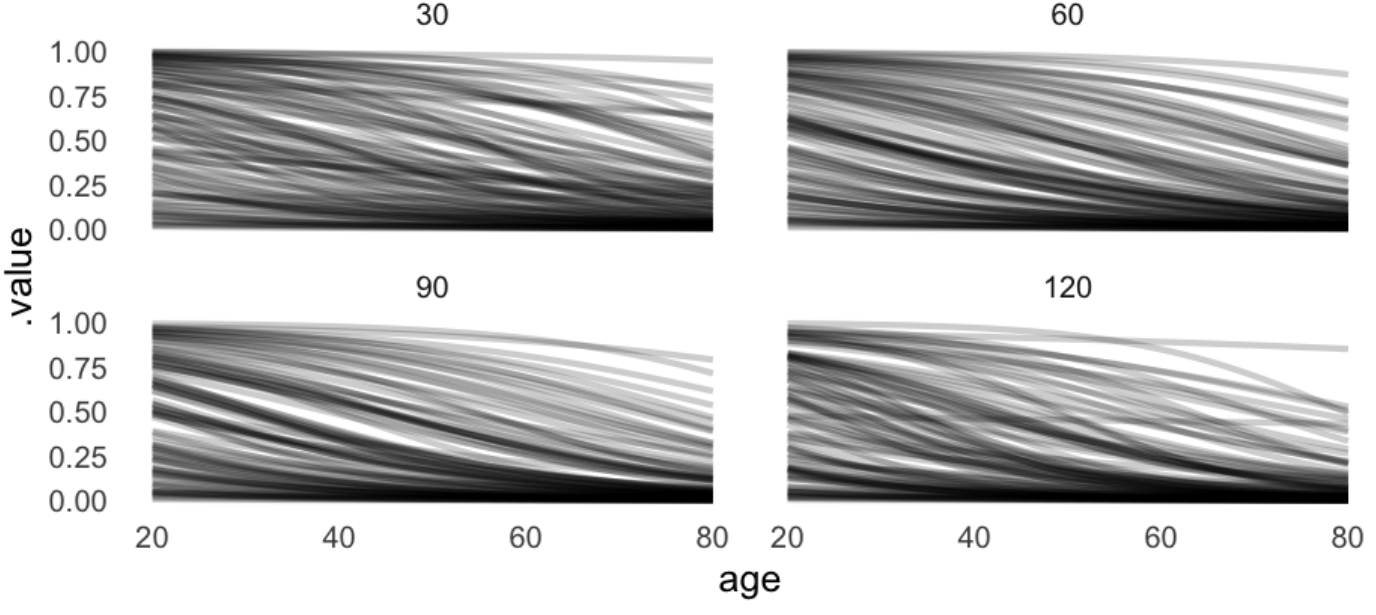
Plot

```
ggplot(pred_tumor, aes(age, .value)) +
  stat_lineribbon()
```



Different plot

```
pred_tumor %>%
  filter(tumorsize %in% c(30, 60, 90, 120)) %>%
ggplot(aes(age, .value)) +
  geom_line(aes(group = .draw), alpha = 0.2) +
  facet_wrap(~tumorsize) +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```



Variance by Doctor

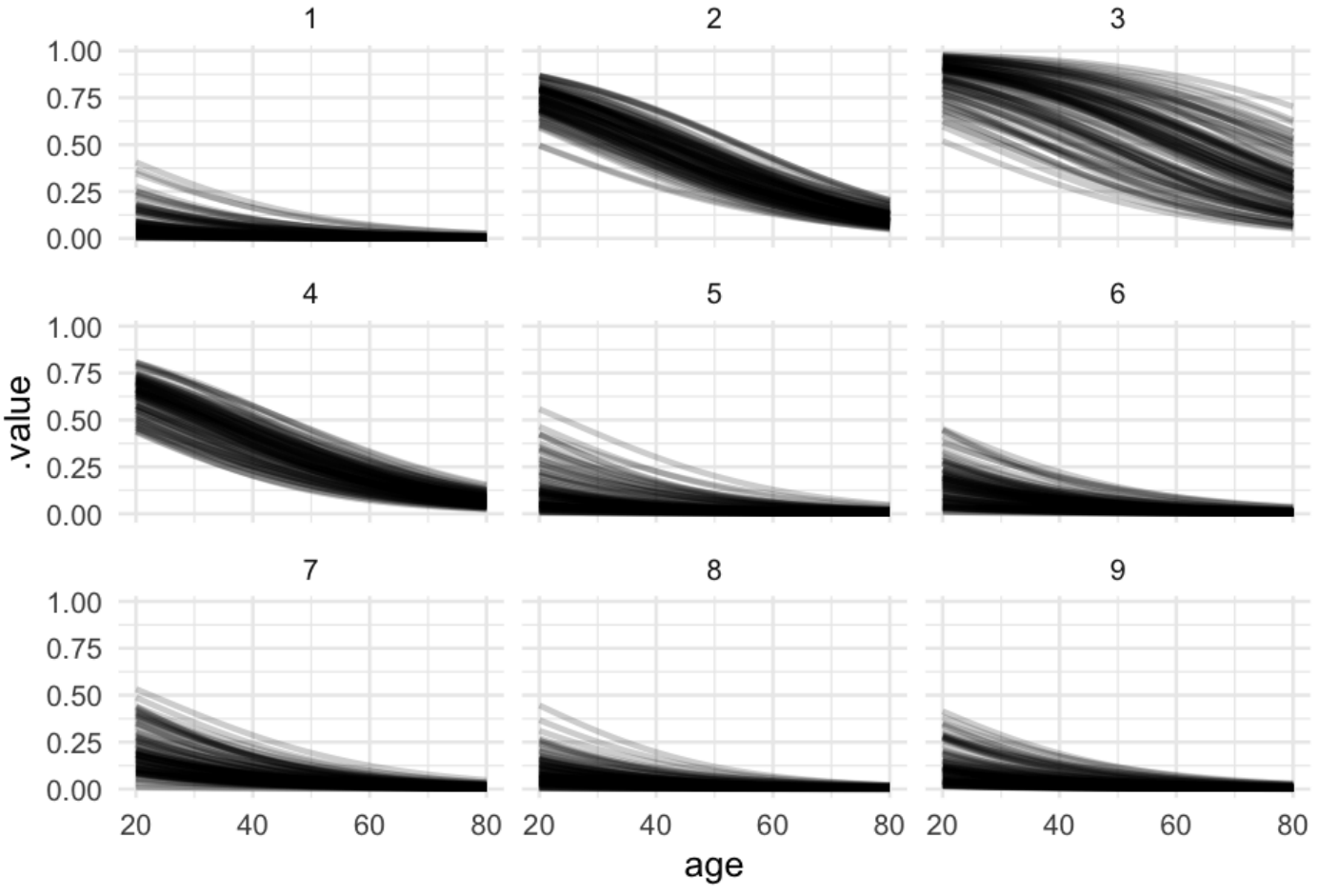
Let's look at the relation between age and proability of remission for each of the first nine doctors.

```
pred_age_doctor <- expand.grid(
    did = unique(hdp$did)[1:9],
    age = 20:80,
    tumorsize = mean(hdp$tumorsize),
    lungcapacity = mean(hdp$lungcapacity)
  ) %>%
  add_fitted_draws(model = lc, n = 100)
```

```
pred_age_doctor
```

```
## # A tibble: 54,900 x 9
## # Groups:   did, age, tumorsize, lungcapacity, .row [549]
##      did   age tumorsize lungcapacity  .row .chain .iteration .draw
##    <dbl> <int>     <dbl>        <dbl> <int>  <int>      <int> <int>
##  1     1    20  70.88067    0.7740865     1     NA         NA    38 0.34
##  2     1    20  70.88067    0.7740865     1     NA         NA    73 0.11
##  3     1    20  70.88067    0.7740865     1     NA         NA    99 0.00
##  4     1    20  70.88067    0.7740865     1     NA         NA   107 0.02
##  5     1    20  70.88067    0.7740865     1     NA         NA   217 0.10
##  6     1    20  70.88067    0.7740865     1     NA         NA   241 0.05
##  7     1    20  70.88067    0.7740865     1     NA         NA   331 0.15
##  8     1    20  70.88067    0.7740865     1     NA         NA   348 0.36
##  9     1    20  70.88067    0.7740865     1     NA         NA   356 0.04
## 10     1    20  70.88067    0.7740865     1     NA         NA   368 0.01
## # … with 54,890 more rows
```

```
ggplot(pred_age_doctor, aes(age, .value)) +
  geom_line(aes(group = .draw), alpha = 0.2) +
  facet_wrap(~did)
```



Look at our variables

```
get_variables(lc)
```

```
##   [1] "b_Intercept"           "b_age"                 "b_tumorsize"
##   [7] "Intercept"             "r_did[1,Intercept]"    "r_did[2,Intercept]"
##  [13] "r_did[6,Intercept]"    "r_did[7,Intercept]"    "r_did[8,Intercept]"
##  [19] "r_did[12,Intercept]"   "r_did[13,Intercept]"   "r_did[14,Intercept]
##  [25] "r_did[18,Intercept]"   "r_did[19,Intercept]"   "r_did[20,Intercept]
##  [31] "r_did[24,Intercept]"   "r_did[25,Intercept]"   "r_did[26,Intercept]
##  [37] "r_did[30,Intercept]"   "r_did[31,Intercept]"   "r_did[32,Intercept]
##  [43] "r_did[36,Intercept]"   "r_did[37,Intercept]"   "r_did[38,Intercept]
##  [49] "r_did[42,Intercept]"   "r_did[43,Intercept]"   "r_did[44,Intercept]
##  [55] "r_did[48,Intercept]"   "r_did[49,Intercept]"   "r_did[50,Intercept]
##  [61] "r_did[54,Intercept]"   "r_did[55,Intercept]"   "r_did[56,Intercept]
##  [67] "r_did[60,Intercept]"   "r_did[61,Intercept]"   "r_did[62,Intercept]
##  [73] "r_did[66,Intercept]"   "r_did[67,Intercept]"   "r_did[68,Intercept]
##  [79] "r_did[72,Intercept]"   "r_did[73,Intercept]"   "r_did[74,Intercept]
##  [85] "r_did[78,Intercept]"   "r_did[79,Intercept]"   "r_did[80,Intercept]
##  [91] "r_did[84,Intercept]"   "r_did[85,Intercept]"   "r_did[86,Intercept]
##  [97] "r_did[90,Intercept]"   "r_did[91,Intercept]"   "r_did[92,Intercept]
## [103] "r_did[96,Intercept]"   "r_did[97,Intercept]"   "r_did[98,Intercept]
## [109] "r_did[102,Intercept]" "r_did[103,Intercept]" "r_did[104,Intercept
## [115] "r_did[108,Intercept]" "r_did[109,Intercept]" "r_did[110,Intercept
## [121] "r_did[114,Intercept]" "r_did[115,Intercept]" "r_did[116,Intercept
## [127] "r_did[120,Intercept]" "r_did[121,Intercept]" "r_did[122,Intercept
## [133] "r_did[126,Intercept]" "r_did[127,Intercept]" "r_did[128,Intercept
## [139] "r_did[132,Intercept]" "r_did[133,Intercept]" "r_did[134,Intercept
```

Get all draws: Intercept

Notice I'm using `spread_draws()` here for a slightly different output format

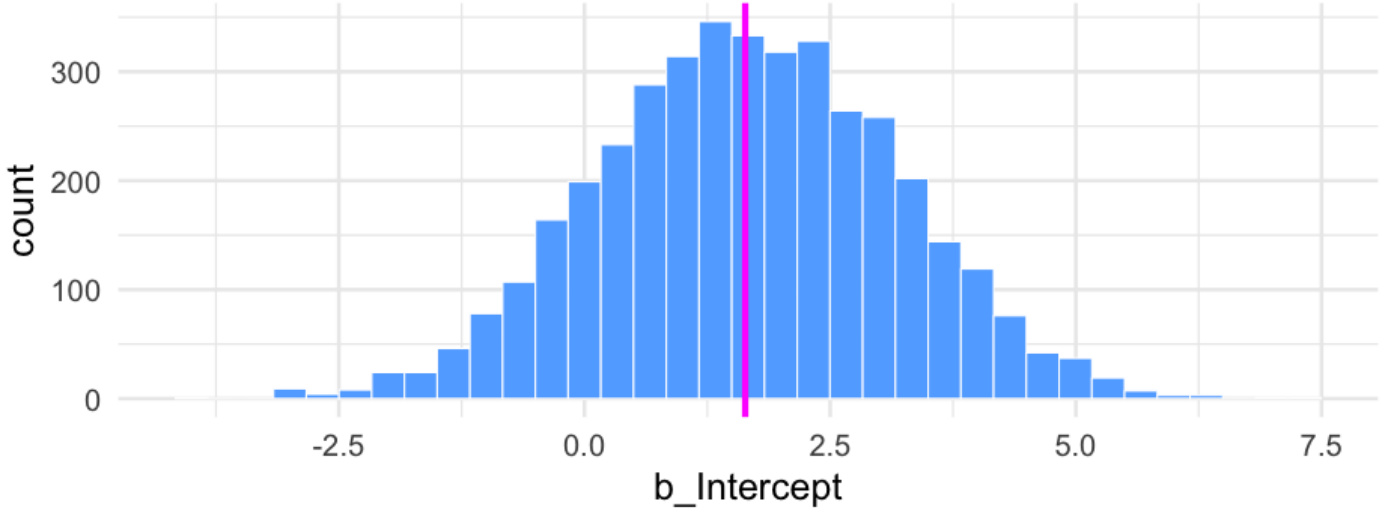
```
int <- lc %>%
  spread_draws(b_Intercept)
int
```

```
## # A tibble: 4,000 x 4
##    .chain .iteration .draw b_Intercept
##     <int>      <int> <int>       <dbl>
##  1      1          1     1    0.799584
##  2      1          2     2   -0.0137843
##  3      1          3     3    1.11889
##  4      1          4     4    2.99813
##  5      1          5     5    1.37627
##  6      1          6     6    2.57053
##  7      1          7     7    3.96603
##  8      1          8     8    3.98211
##  9      1          9     9    0.561609
## 10      1         10    10    0.135172
## # … with 3,990 more rows
```

Plot the distribution

Alternative to `insight::get_parameters()`

```
ggplot(int, aes(b_Intercept)) +
  geom_histogram(fill = "#61adff",
                 color = "white",
                 bins = 35) +
  geom_vline(aes(xintercept = median(b_Intercept)),
             color = "magenta",
             size = 2)
```



Grab random effects

- The random effect name is `r_did`

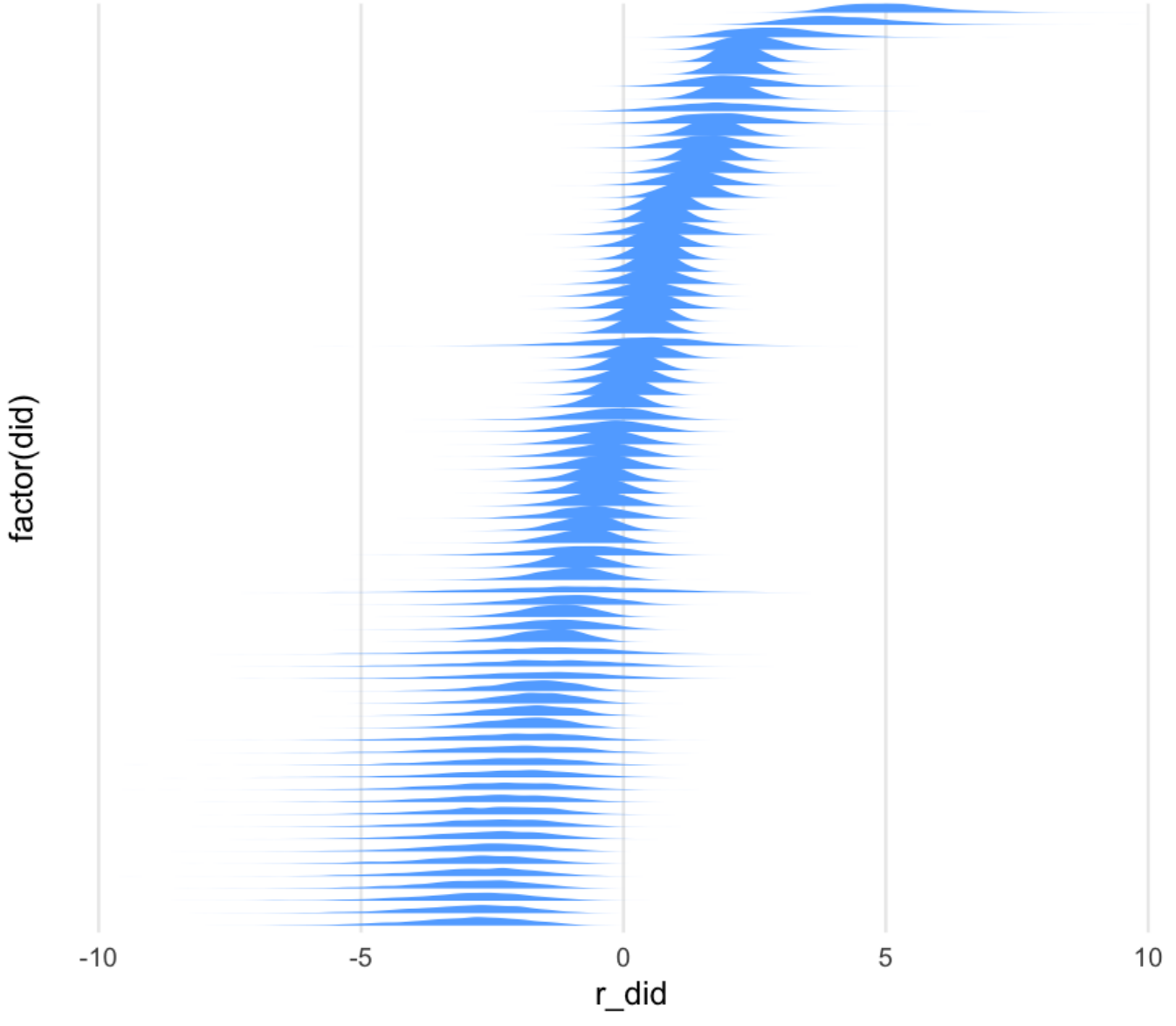
```
spread_draws(lc, r_did[did, term])
```

```
## # A tibble: 1,628,000 x 6
## # Groups:   did, term [407]
##      did term          r_did .chain .iteration .draw
##    <int> <chr>         <dbl>  <int>      <int> <int>
##  1     1 Intercept -1.94541      1          1     1
##  2     1 Intercept -8.03429      1          2     2
##  3     1 Intercept -3.57734      1          3     3
##  4     1 Intercept -2.27834      1          4     4
##  5     1 Intercept -3.41082      1          5     5
##  6     1 Intercept -2.17562      1          6     6
##  7     1 Intercept -3.63758      1          7     7
##  8     1 Intercept -2.9472       1          8     8
##  9     1 Intercept -2.66865      1          9     9
## 10     1 Intercept -5.06383      1         10    10
## # … with 1,627,990 more rows
```

Look at did distributions

First 75 doctors

```
dids <- spread_draws(lc, r_did[did, ]) # all terms, which is just
dids %>%
  ungroup() %>%
  filter(did %in% 1:75) %>%
  mutate(did = fct_reorder(factor(did), r_did)) %>%
  ggplot(aes(x = r_did, y = factor(did))) +
  ggridges::geom_density_ridges(color = NA, fill = "#61adff") +
  theme(panel.grid.major.y = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.y = element_blank())
```


factor(did)
-10
-5
0
5
10
r_did

Long format

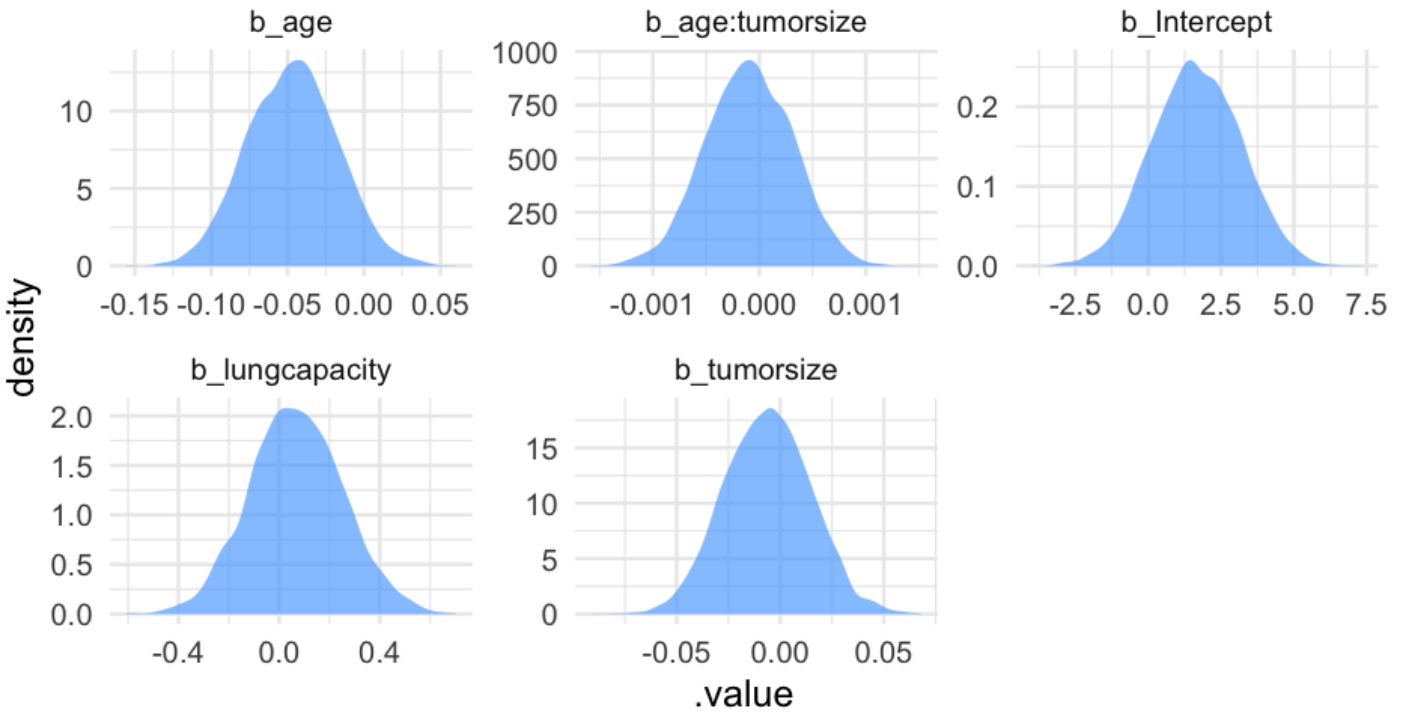
Use `gather_draws()` for a longer format (as we did before)

```
fixed_l <- lc %>%
  gather_draws(b_Intercept, b_age, b_tumorsize, b_lungcapacity,
               `b_age:tumorsize`)
fixed_l
```

```
## # A tibble: 20,000 x 5
## # Groups:   .variable [5]
##    .chain .iteration .draw .variable       .value
##     <int>      <int> <int> <chr>            <dbl>
##  1      1          1     1 b_Intercept  0.799584
##  2      1          2     2 b_Intercept -0.0137843
##  3      1          3     3 b_Intercept  1.11889
##  4      1          4     4 b_Intercept  2.99813
##  5      1          5     5 b_Intercept  1.37627
##  6      1          6     6 b_Intercept  2.57053
##  7      1          7     7 b_Intercept  3.96603
##  8      1          8     8 b_Intercept  3.98211
##  9      1          9     9 b_Intercept  0.561609
## 10      1         10    10 b_Intercept  0.135172
## # … with 19,990 more rows
```

Plot the densities

```
ggplot(fixed_l, aes(.value)) +
  geom_density(fill = "#61adff", alpha = 0.7, color = NA) +
  facet_wrap(~.variable, scales = "free")
```



Multiple comparisons

One of the nicest things about Bayes is that any comparison you want to make can be made without jumping through a lot of additional hoops (e.g., adjusting α).

Scenario

Imagine a **35** year old has a tumor measuring **58 millimeters** and a lung capacity rating of **0.81**.

What would we estimate as the probability of remission if this patient had `did == 1` versus `did == 2`?

Fixed effects

Not really "fixed", but rather just average relation

```
fe <- lc %>%
  spread_draws(b_Intercept, b_age, b_tumorsize, b_lungcapacity,
               `b_age:tumorsize`)
fe
```

```
## # A tibble: 4,000 x 8
##    .chain .iteration .draw b_Intercept      b_age b_tumorsize b_lungcapa
##     <int>      <int> <int>       <dbl>      <dbl>       <dbl>       <
##  1      1          1     1   0.799584  -0.0318583  0.00418452     0.1792
##  2      1          2     2  -0.0137843 -0.016284   0.0151118      0.0433
##  3      1          3     3   1.11889   -0.0304524  0.00133416     0.0474
##  4      1          4     4   2.99813   -0.0769063 -0.0199561      0.1573
##  5      1          5     5   1.37627   -0.0447789  0.00165146     0.2046
##  6      1          6     6   2.57053   -0.0661329 -0.0211792      0.1617
##  7      1          7     7   3.96603   -0.0934501 -0.0318102      0.1216
##  8      1          8     8   3.98211   -0.090909  -0.0310474     -0.0166
##  9      1          9     9   0.561609  -0.0246561  0.00422727     0.1612
## 10      1         10    10   0.135172  -0.0192131  0.0167028     -0.0024
## # … with 3,990 more rows
```

Data

```
age <- 35
tumor_size <- 58
lung_cap <- 0.81
```

Population–level predictions (there's other ways we could do this, but it's good to remind ourselves the "by hand" version too)

```
pop_level <-
  fe$b_Intercept +
  (fe$b_age * age) +
  (fe$b_tumorsize * tumor_size) +
  (fe$b_lungcapacity * lung_cap) +
  (fe$`b_age:tumorsize` * (age * tumor_size))
pop_level
```

```
##    [1] -0.49463415 -0.63799719 -0.45421805 -0.30701290 -0.36786178 -0.43
##   [13] -0.51347752 -0.35731807 -0.30377370 -0.45074384 -0.39412555 -0.42
##   [25] -0.41777911 -0.24052536 -0.26502943 -0.14023521 -0.08610160 -0.25
##   [37] -0.39127941 -0.54278890 -0.81346645 -0.82226173 -0.69583974 -0.50
```

Plot population level

```
pd <- tibble(population_level = pop_level)

ggplot(pd, aes(population_level)) +
  geom_histogram(fill = "#61adff",
                 color = "white") +
  geom_vline(xintercept = median(pd$population_level),
             color = "magenta",
             size = 2)
```

Add in did estimates

```
did1 <- filter(dids, did == 1)
did2 <- filter(dids, did == 2)

pred_did1 <- pop_level + did1$r_did
pred_did2 <- pop_level + did2$r_did
```

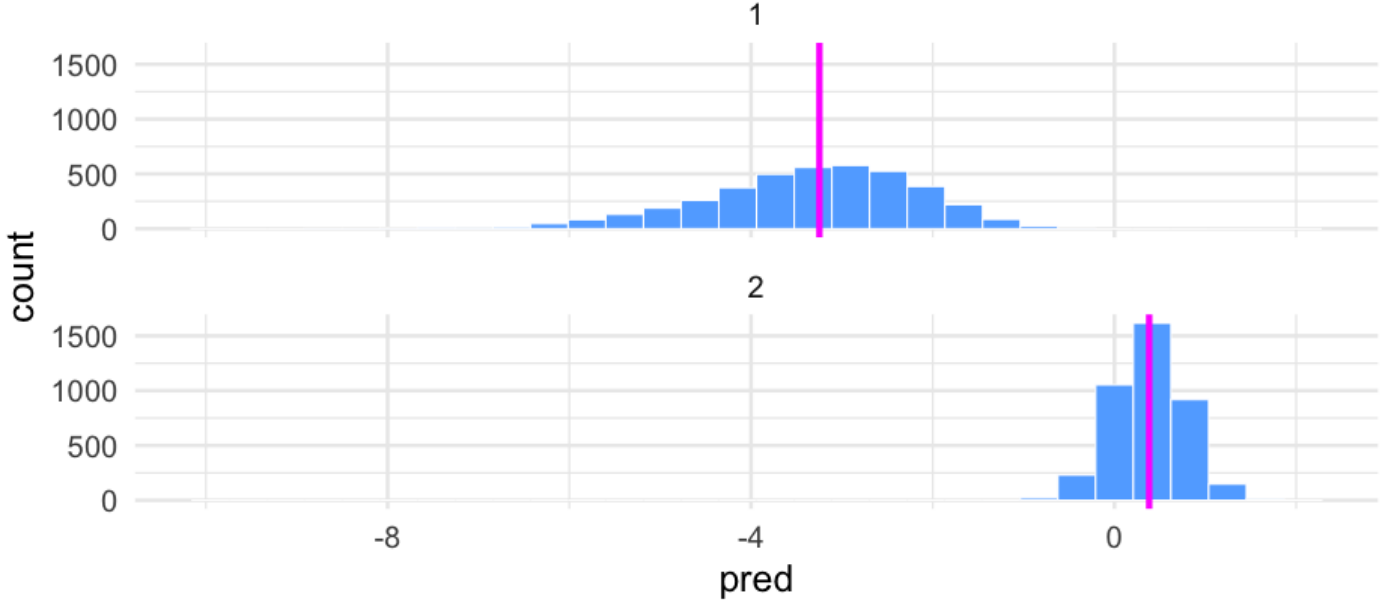
Distributions

```
did12 <- tibble(did = rep(1:2, each = length(pred_did1)),
                pred = c(pred_did1, pred_did2))

did12_medians <- did12 %>%
  group_by(did) %>%
  summarize(did_median = median(pred))

did12_medians
```

```
## # A tibble: 2 x 2
##     did did_median
##   <int>      <dbl>
## 1     1 -3.247639
## 2     2  0.3819962
```

Plot

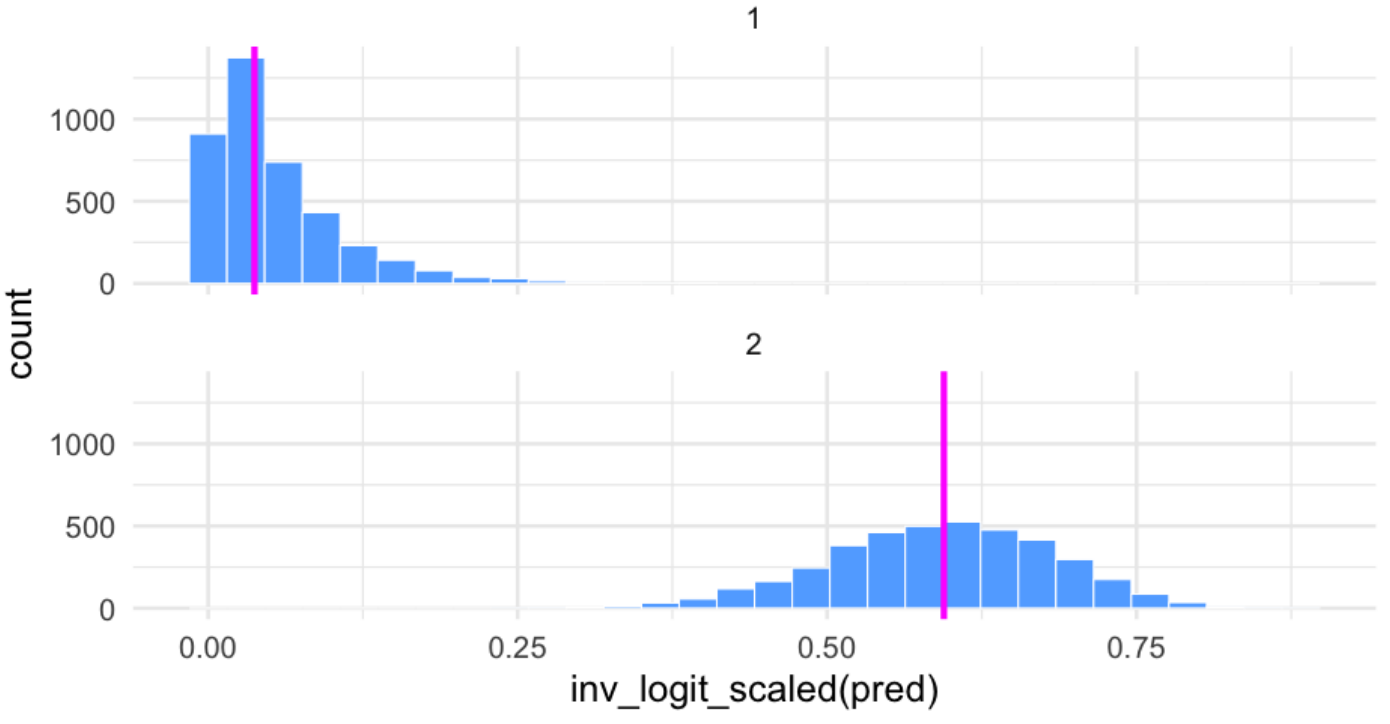
```
ggplot(did12, aes(pred)) +
  geom_histogram(fill = "#61adff",
                 color = "white") +
  geom_vline(aes(xintercept = did_median), data = did12_medians,
             color = "magenta",
             size = 2) +
  facet_wrap(~did, ncol = 1)
```



Transform

Let's look at this again on the probability scale using `brms::inv_logit_scaled()` to make the transformation.

```
ggplot(did12, aes(inv_logit_scaled(pred))) +
  geom_histogram(fill = "#61adff",
                 color = "white") +
  geom_vline(aes(xintercept = inv_logit_scaled(did_median)),
             data = did12_medians,
             color = "magenta",
             size = 2) +
  facet_wrap(~did, ncol = 1)
```


1
2
1000
500
0
count
0.00
0.25
0.50
0.75
inv_logit_scaled(pred)

Difference

- The difference in the probability of remission for our theoretical patient is large between the two doctors.
- The median difference in log–odds is

```
diff(did12_medians$did_median)
```

```
## [1] 3.629635
```

Exponentiation

We can exponentiate the log–odds to get normal odds

These are fairly interpretable (especially when greater than 1)

```
# probability
inv_logit_scaled(did12_medians$did_median)
```

```
## [1] 0.03741181 0.59435448
```

```
# odds
exp(did12_medians$did_median)
```

```
## [1] 0.03886586 1.46520658
```

```
# odds of the difference
exp(diff(did12_medians$did_median))
```

```
## [1] 37.69907
```

We estimate that our theoretical patient is about 38 times **more likely** (!) to go into remission if they had `did` 2, instead of 1.

Confidence in difference?

Everything is a distribution

Just compute the difference in these distributions, and we get a new distribution, which we can use to summarize our uncertainty

```
did12_wider <- tibble(
  did1 = pred_did1,
  did2 = pred_did2
) %>%
  mutate(diff = did2 - did1)

did12_wider
```

```
## # A tibble: 4,000 x 3
##         did1       did2     diff
##        <dbl>      <dbl>    <dbl>
##  1 -2.440044  0.7427359 3.18278
##  2 -8.672287 -0.2324662 8.439821
##  3 -4.031558  0.2963330 4.327891
##  4 -2.585353  0.8626371 3.44799
```

Summarize

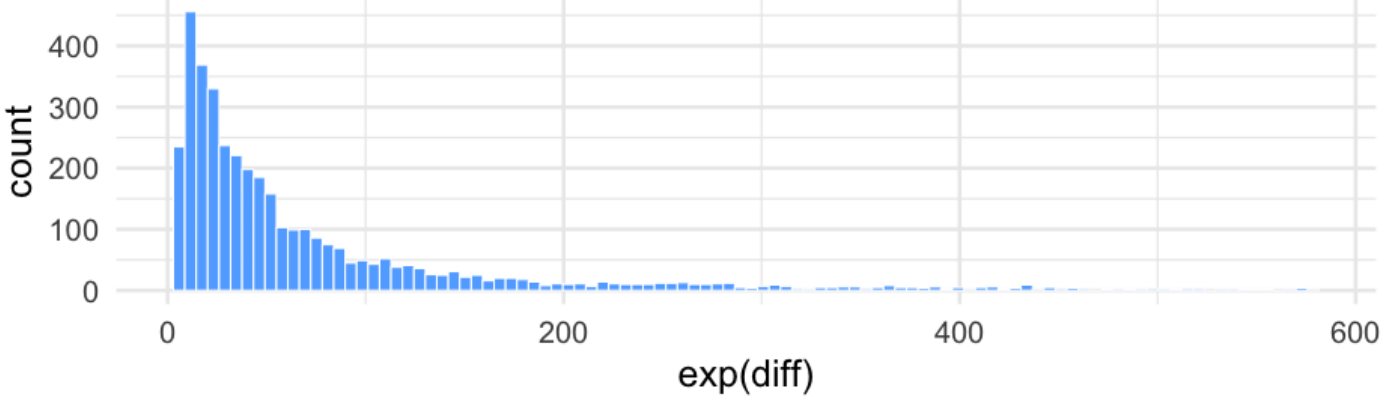
```
qtile_diffs <- quantile(did12_wider$diff,
                        probs = c(0.025, 0.5, 0.975))

exp(qtile_diffs)
```

```
##       2.5%        50%      97.5%
##   5.388731  38.962124 578.302883
```

Plot distribution

Show the most likely 95% of the distribution

```
# filter a few extreme observations
did12_wider %>%
  filter(diff > qtile_diffs[1] & diff < qtile_diffs[3]) %>%
ggplot(aes(exp(diff))) +
  geom_histogram(fill = "#61adff",
                 color = "white",
                 bins = 100)
```



Directionality

Let's say we want to simplify the question to directionality.

Is there a greater chance of remission for `did` 2 than 1?

```
table(did12_wider$diff > 0) / 4000
```

```
##
## TRUE
##    1
```

The distributions are not overlapping at all – therefore, we are as certain as we can be that the odds of remission are higher with `did` 2 than 1.

One more quick example

Let's do the same thing, but comparing `did` 2 and 3.

```
did3 <- filter(dids, did == 3)
pred_did3 <- pop_level + did3$r_did

did23 <- did12_wider %>%
  select(-did1, -diff) %>%
  mutate(did3 = pred_did3,
         diff = did3 - did2)
did23
```

```
## # A tibble: 4,000 x 3
##          did2     did3     diff
##         <dbl>    <dbl>    <dbl>
##  1  0.7427359 2.316826 1.57409
##  2 -0.2324662 1.038893 1.271359
##  3  0.2963330 1.632892 1.336559
##  4  0.8626371 1.552087 0.68945
##  5  0.2879902 1.575678 1.287688
##  6  0.2724074 1.483903 1.211496
##  7  0.7147088 1.749563 1.034854
##  8  0.6158916 1.229078 0.613186
##  9  0.4442302 1.474110 1.02988
## 10  0.2998222 1.705350 1.405528
```

Directionality

```
table(did23$diff > 0) / 4000
```

```
##
##   FALSE    TRUE
## 0.13425 0.86575
```

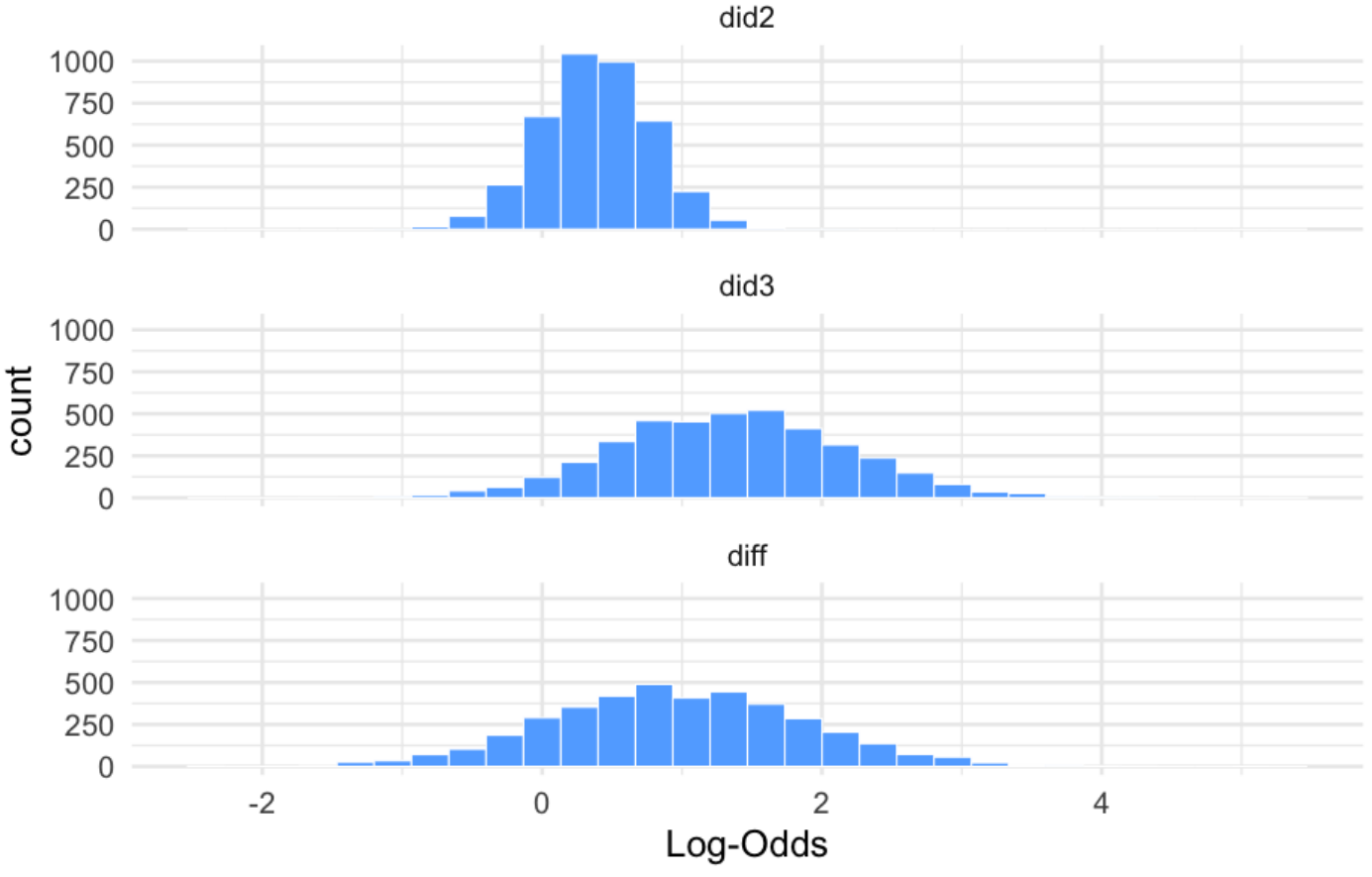
So there's roughly an 87% chance that the odds of remission are higher with with `did` 3 than 2.

Plot data

```
pd23 <- did23 %>%
  pivot_longer(did2:diff,
               names_to = "Distribution",
               values_to = "Log-Odds")
pd23
```

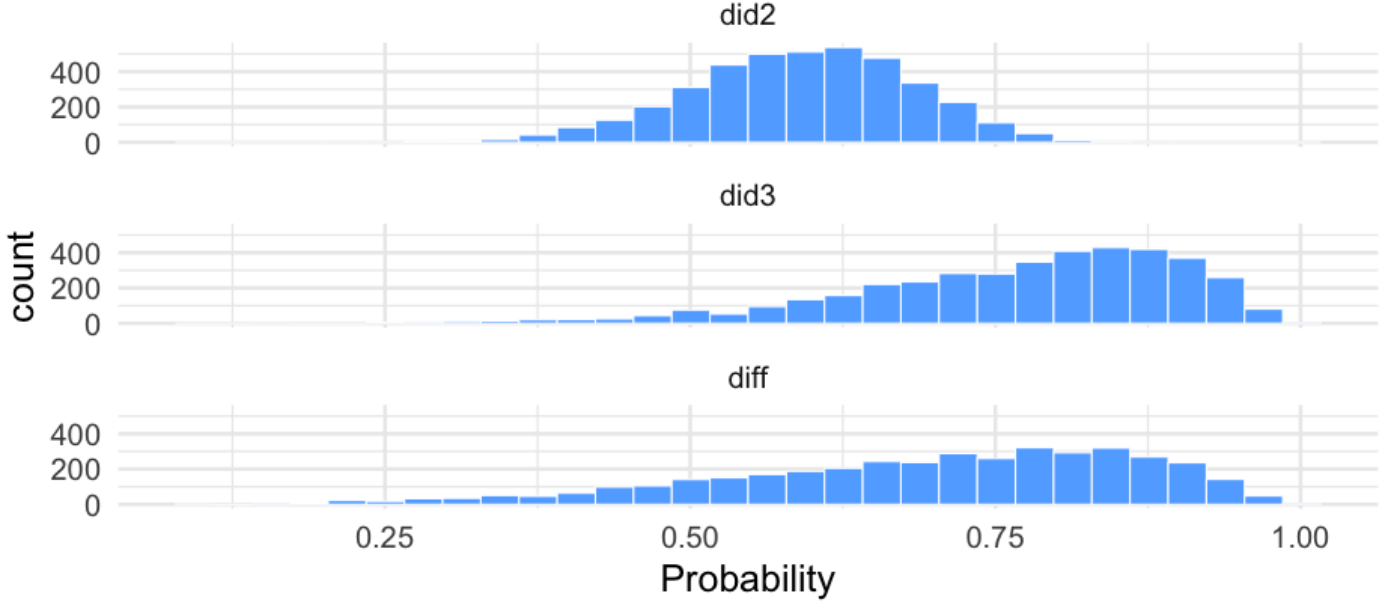
```
## # A tibble: 12,000 x 2
##    Distribution `Log-Odds`
##    <chr>             <dbl>
##  1 did2          0.7427359
##  2 did3          2.316826
##  3 diff          1.57409
##  4 did2         -0.2324662
##  5 did3          1.038893
##  6 diff          1.271359
##  7 did2          0.2963330
##  8 did3          1.632892
##  9 diff          1.336559
## 10 did2          0.8626371
## # … with 11,990 more rows
```

```
ggplot(pd23, aes(`Log-Odds`)) +
  geom_histogram(fill = "#61adff",
                 color = "white") +
  facet_wrap(~Distribution, ncol = 1)
```



Probability scale

```
pd23 %>%
  mutate(Probability = inv_logit_scaled(`Log-Odds`)) %>%
ggplot(aes(Probability)) +
  geom_histogram(fill = "#61adff",
                 color = "white") +
  facet_wrap(~Distribution, ncol = 1)
```



Any time left?

Missing data

Disclaimer

- Missing data is a **massive** topic
- I'm hoping/assuming you've covered it some in other classes
- This is mostly about implementation options

Missing data on the DV

- Mostly what we tend to talk about in classes
- Also regularly the least problematic
- If we can assume MAR (missing at random conditional on covariates), most modern models do a pretty good job

Missing data on the IVs

- Much more problematic, no matter the model or application
- Remove all cases with any missingness on any IV?
    - Limits your sample size
    - Might (probably?) introduces new sources of bias
- Impute?
    - Often ethical challenges here – do you really want to impute somebody's gender?

Solution?

- There really isn't a great one. Be clear about the decisions you do make.
- If you do choose imputation, use multiple imputation
    - This will allow you to have uncertainty in your imputation
- The purpose is to get unbiased population estimates for your parameters (not make inferences about an individual for whom data were imputed)

Missing IDs

- In multilevel models, you always have IDs linking the data to the higher levels
- If you are missing these IDs, I'm not really sure what to tell you
    - This is particularly common with longitudinal data (e.g., missing prior school IDs)
    - In rare cases, you can make assumptions and impute, but those are few and far between, in my experience, and the assumptions are still pretty dangerous

Let's do it

Multiple imputation

This part is general, and not specific to multilevel modeling

First, install/load the **{mice}** package (you might also check out **{Amelia}**)

```
library(mice)
```

Data

We'll impute data from the nhanes dataset, which comes with **{mice}**

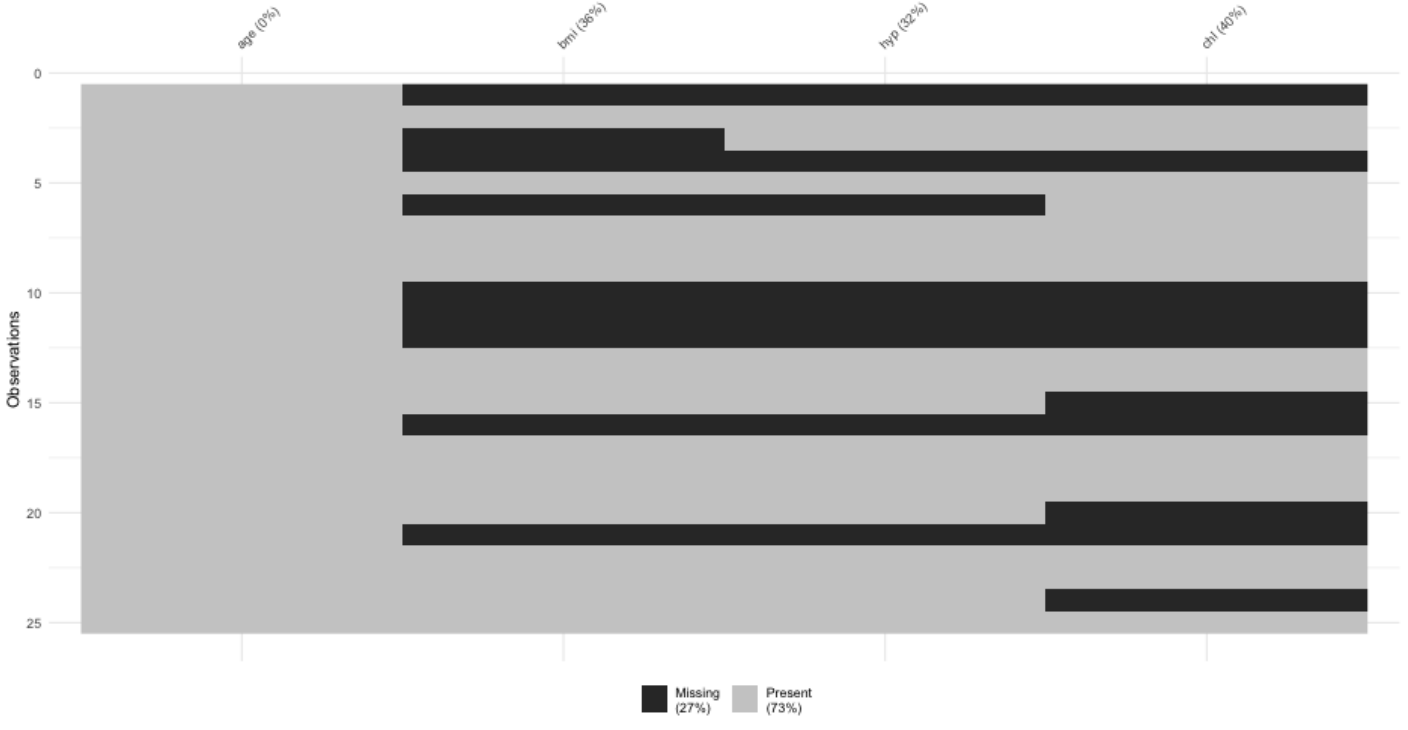
```
head(nhanes)
```

```
##   age  bmi hyp chl
## 1   1   NA  NA  NA
## 2   2 22.7   1 187
## 3   1   NA   1 187
## 4   3   NA  NA  NA
## 5   1 20.4   1 113
## 6   3   NA  NA 184
```

How much missingness

A lot

```
#install.packages("naniar")
naniar::vis_miss(nhanes)
```



Multiple imputation

- First, we're going to create 5 new dataset, each one with the missing data imputed

```
mi_nhanes <- mice(nhanes, m = 5, print = FALSE)
```

MI for BMI

```
mi_nhanes$imp$bmi
```

```
##       1    2    3    4    5
## 1  30.1 27.2 33.2 29.6 30.1
## 3  30.1 30.1 29.6 26.3 30.1
## 4  21.7 22.5 27.4 22.5 25.5
## 6  24.9 24.9 21.7 21.7 25.5
## 10 28.7 27.4 27.2 20.4 27.4
## 11 30.1 28.7 22.0 28.7 35.3
## 12 27.5 22.0 22.5 27.4 28.7
## 16 35.3 27.2 27.2 26.3 30.1
## 21 29.6 22.0 22.0 26.3 33.2
```

Fit model w/brms

Now just feed the **{mice}** object to `brms::brm_multiple()` as your data.

Note – this is considerably easier than it is with **lme4**, but it is do–able

```
m_mice <- brm_multiple(bmi ~ age * chl,
                       data = mi_nhanes)
```

Alternative

- A neat thing we can do with Bayes, is to impute *on the fly* using the posterior
- We still get uncertainty because of the repeated samples we're taking from the posterior anyway
- With **{brms}**, we can do this by just passing a slightly more complicated formula

Missing formula

We specify a model for each column that has missingness

We have missing data in `bmi` and `chl` (not `age`).

`bmi` is our outcome, and it will be modeled by `age` and the *complete* (missing data imputed) `chl` variable, as well as their interaction

The missing data in `chl` will be imputed via a model with `age` as its predictor!

We're basically fitting two models at once.

In code

The `| mi()` part says to include missing data, while `

```
bayes_impute_formula <- bf(bmi | mi() ~ age * mi(chl)) + # base r
  bf(chl | mi() ~ age) + # model for chl missingness
  set_rescor(FALSE) # we don't estimate the residual correlation
```

Fit

```
m_onfly <- brm(bayes_impute_formula, data = nhanes)
```

Comparison

Multiple imputation before modeling

Next time

Piece–wise models, cross–classification & (maybe) multiple membership models