

More Bayes

And multilevel binomial logistic
regression

Daniel Anderson

Week 8

Agenda

- More equation practice
- Finishing up our intro to Bayes
 - Specific focus on MCMC
- Implementation with **{brms}**
- Logistic regression review
- Extending to multilevel logistic regression models with **lme4**

Equation

practice

The data

Read in the following data

```
library(tidyverse)
library(lme4)
d <- read_csv(here::here("data", "three-lev.csv"))
d
```

```
## # A tibble: 7,230 x 12
##   schid      sid  size lowinc mobility female black hispanic retained
##   <dbl>    <dbl> <dbl>  <dbl>    <dbl>   <dbl> <dbl>    <dbl>    <dbl>
## 1  2020 273026452   380   40.3     12.5     0     0         1         0
## 2  2020 273026452   380   40.3     12.5     0     0         1         0
## 3  2020 273026452   380   40.3     12.5     0     0         1         0
## 4  2020 273030991   380   40.3     12.5     0     0         0         0
## 5  2020 273030991   380   40.3     12.5     0     0         0         0
## 6  2020 273030991   380   40.3     12.5     0     0         0         0
## 7  2020 273030991   380   40.3     12.5     0     0         0         0
## 8  2020 273030991   380   40.3     12.5     0     0         0         0
## 9  2020 273059461   380   40.3     12.5     0     0         1         0
## 10 2020 273059461   380   40.3     12.5     0     0         1         0
## # ... with 7,220 more rows, and 2 more variables: year <dbl>, math <dbl>
```

Model 1

Fit the following model

$$\begin{aligned}\text{math}_i &\sim N(\alpha_{j[i]}, \sigma^2) \\ \alpha_j &\sim N(\gamma_0^\alpha + \gamma_1^\alpha(\text{mobility}), \sigma_{\alpha_j}^2), \text{ for schid } j = 1, \dots, J\end{aligned}$$

```
lmer(math ~ mobility + (1|schid), data = d)
```

01:30

Model 2

Fit the following model

$$\begin{aligned}\text{math}_i &\sim N(\alpha_{j[i],k[i]} + \beta_{1j[i]}(\text{year}), \sigma^2) \\ \begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N \left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{female}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix} \right), \text{ for sid } j = 1, \dots, J \\ \alpha_k &\sim N(\gamma_0^\alpha + \gamma_1^\alpha(\text{mobility}), \sigma_{\alpha_k}^2), \text{ for schid } k = 1, \dots, K\end{aligned}$$

```
lmer(math ~ year + female + mobility +  
      (year|sid) + (1|schid),  
      data = d)
```

01:30

Model 3

Fit the following model

$$\begin{aligned} \text{math}_i &\sim N(\alpha_{j[i],k[i]} + \beta_{1j[i]}(\text{year}), \sigma^2) \\ \begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N \left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{black}) + \gamma_2^\alpha(\text{hispanic}) + \gamma_3^\alpha(\text{female}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{black}) + \gamma_2^{\beta_1}(\text{hispanic}) \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix} \right), \text{ for sid } j \\ \alpha_k &\sim N(\gamma_0^\alpha + \gamma_1^\alpha(\text{mobility}), \sigma_{\alpha_k}^2), \text{ for schid } k = 1, \dots, K \end{aligned}$$

```
lmer(math ~ year * black + year * hispanic + female + mobility +  
      (year|sid) + (1|schid),  
      data = d)
```

01:30

Model 4

Fit the following model

$$\begin{aligned} \text{math}_i &\sim N(\alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{year}), \sigma^2) \\ \begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N \left(\begin{pmatrix} \gamma_0^\alpha + \gamma_{1k[i]}^\alpha(\text{female}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{female}) \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix} \right), \text{ for sid } j = 1, \dots, J \\ \begin{pmatrix} \alpha_k \\ \beta_{1k} \\ \gamma_{1k} \end{pmatrix} &\sim N \left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{mobility}) + \gamma_2^\alpha(\text{lowinc}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{mobility}) \\ \mu_{\gamma_{1k}} \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_k}^2 & \rho_{\alpha_k\beta_{1k}} & \rho_{\alpha_k\gamma_{1k}} \\ \rho_{\beta_{1k}\alpha_k} & \sigma_{\beta_{1k}}^2 & \rho_{\beta_{1k}\gamma_{1k}} \\ \rho_{\gamma_{1k}\alpha_k} & \rho_{\gamma_{1k}\beta_{1k}} & \sigma_{\gamma_{1k}}^2 \end{pmatrix} \right), \text{ for schid } k \end{aligned}$$

```
lmer(math ~ year * female + year * mobility + lowinc +
      (year|sid) + (year + female|schid),
      data = d)
```

01:30

Model 5

Fit the following model

Don't worry if you run into convergence warnings

$$\begin{aligned} \text{math}_i &\sim N(\alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{year}), \sigma^2) \\ \begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N \left(\begin{pmatrix} \gamma_0^\alpha + \gamma_{1k[i]}^\alpha(\text{female}) \\ \gamma_{1k[i]}^{\beta_1} + \gamma_{1k[i]}^{\beta_1}(\text{female}) \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix} \right), \text{ for sid } j = 1, \dots, J \\ \begin{pmatrix} \alpha_k \\ \beta_{1k} \\ \gamma_{1k} \\ \gamma_{1k}^{\beta_1} \end{pmatrix} &\sim N \left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{mobility}) + \gamma_2^\alpha(\text{lowinc}) \\ \mu_{\beta_{1k}} \\ \mu_{\gamma_{1k}} \\ \mu_{\gamma_{1k}^{\beta_1}} \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_k}^2 & 0 & 0 & 0 \\ 0 & \sigma_{\beta_{1k}}^2 & 0 & 0 \\ 0 & 0 & \sigma_{\gamma_{1k}}^2 & 0 \\ 0 & 0 & 0 & \sigma_{\gamma_{1k}^{\beta_1}}^2 \end{pmatrix} \right), \text{ for schid } k = 1, \dots, K \end{aligned}$$

```
lmer(math ~ year * female + mobility + lowinc +  
      (year|sid) + (year * female||schid),  
      data = d)
```

01:30

Model 6

Fit the following model

$$\begin{aligned}\text{math}_i &\sim N(\alpha_{j[i]} + \beta_{1j[i]}(\text{year}), \sigma^2) \\ \begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N \left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{female}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j \beta_{1j}} \\ \rho_{\beta_{1j} \alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix} \right), \text{ for sid } j = 1, \dots, J \\ \gamma_{1k} &\sim N(\mu_{\gamma_{1k}}, \sigma_{\gamma_{1k}}^2), \text{ for schid } k = 1, \dots, K\end{aligned}$$

```
lmer(math ~ year + female + mobility * size +  
      (year|sid) + (0 + mobility|schid),  
      data = d)
```

01:30

Model 7

Fit the following model

$$\text{math}_i \sim N(\alpha_{j[i]} + \beta_{1j[i]}(\text{year}), \sigma^2)$$
$$\begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} \sim N \left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{female}) + \gamma_2^\alpha(\text{mobility}) + \gamma_3^\alpha(\text{size}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j \beta_{1j}} \\ \rho_{\beta_{1j} \alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix} \right), \text{ for sid } j = 1, \dots, J$$

```
lmer(math ~ year + female + mobility + size +  
      (year|sid),  
      data = d)
```

01:30

Model 8

Last one

$$\begin{aligned} \text{math}_i &\sim N(\alpha_{j[i],k[i]} + \beta_{1j[i],k[i]}(\text{year}), \sigma^2) \\ \begin{pmatrix} \alpha_j \\ \beta_{1j} \end{pmatrix} &\sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{female}) \\ \mu_{\beta_{1j}} \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_j}^2 & \rho_{\alpha_j\beta_{1j}} \\ \rho_{\beta_{1j}\alpha_j} & \sigma_{\beta_{1j}}^2 \end{pmatrix}\right), \text{ for sid } j = 1, \dots, J \\ \begin{pmatrix} \alpha_k \\ \beta_{1k} \end{pmatrix} &\sim N\left(\begin{pmatrix} \gamma_0^\alpha + \gamma_1^\alpha(\text{lowinc}) \\ \gamma_0^{\beta_1} + \gamma_1^{\beta_1}(\text{lowinc}) \end{pmatrix}, \begin{pmatrix} \sigma_{\alpha_k}^2 & \rho_{\alpha_k\beta_{1k}} \\ \rho_{\beta_{1k}\alpha_k} & \sigma_{\beta_{1k}}^2 \end{pmatrix}\right), \text{ for schid } k = 1, \dots, K \end{aligned}$$

```
lmer(math ~ year * lowinc + female +  
      (year|sid) + (year |schid),  
      data = d)
```

01:30

Finishing up
w/Bayes intro

Updated beliefs

- Remember our posterior is defined by

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{average likelihood}}$$

When we just had a single parameter to estimate, μ , this was tractable with grid search.

With even simple linear regression, however, we have three parameters: α , β , and σ

Our Bayesian model then becomes considerably more complicated:

$$P(\alpha, \beta, \sigma | x) = \frac{P(x | \alpha, \beta, \sigma) P(\alpha, \beta, \sigma)}{\iiint P(x | \alpha, \beta, \sigma) P(\alpha, \beta, \sigma) d\alpha d\beta d\sigma}$$

Estimation

Rather than trying to compute the integrals, we draw *samples* from the joint posterior distribution.

This sounds a bit like magic – how do we do this?

Multiple different algorithms, but all use some form of Markov–Chain Monte–Carlo sampling

Conceptually

- Imagine the posterior as a hill
- We start with the parameters set to random numbers
 - Estimate the posterior with this values (our spot on the hill)
- Try to "walk" around in a way that we get a complete "picture" of the hill from the samples
- Use these samples as your posterior distribution

An example

Let's go back to an example where we're trying to estimate the mean with a known standard deviation.

$$N(\mu, 15)$$

Start with an initial guess, say 110

MCMC

One algorithm:

- Generate a *proposal* by taking a sample from your best guess according to some distribution, e.g., **$\text{proposal} \sim N(110, 10)$** . Suppose we get a value of 108
- Observe the actual data. Let's say we have only one point: 100
- Compare the "height" of the proposal distribution to height of the current distribution, relative to observed data: **$N(100|110, 15)$, $N(100|108, 15)$**

In code

```
dnorm(110, 100, 15)
```

```
## [1] 0.02129653
```

```
dnorm(108, 100, 15)
```

```
## [1] 0.02307026
```

- If probability of target distribution is higher, accept
- If probability is lower, randomly select between the two with probability equal to the "heights" (probability of the two distributions)
- If proposal is accepted – it's the next sample in the chain
- Otherwise, the next sample is a copy of current sample

This completes one iteration

Next iteration starts by generating a new proposal distribution

Stop when you have enough samples to have an adequate "picture"

Example in code

Let's take 5000 samples

```
set.seed(42) # for reproducibility

samples <- c(
  110, # initial guess
  rep(NA, 4999) # space for subsequent samples to fill in
)
```

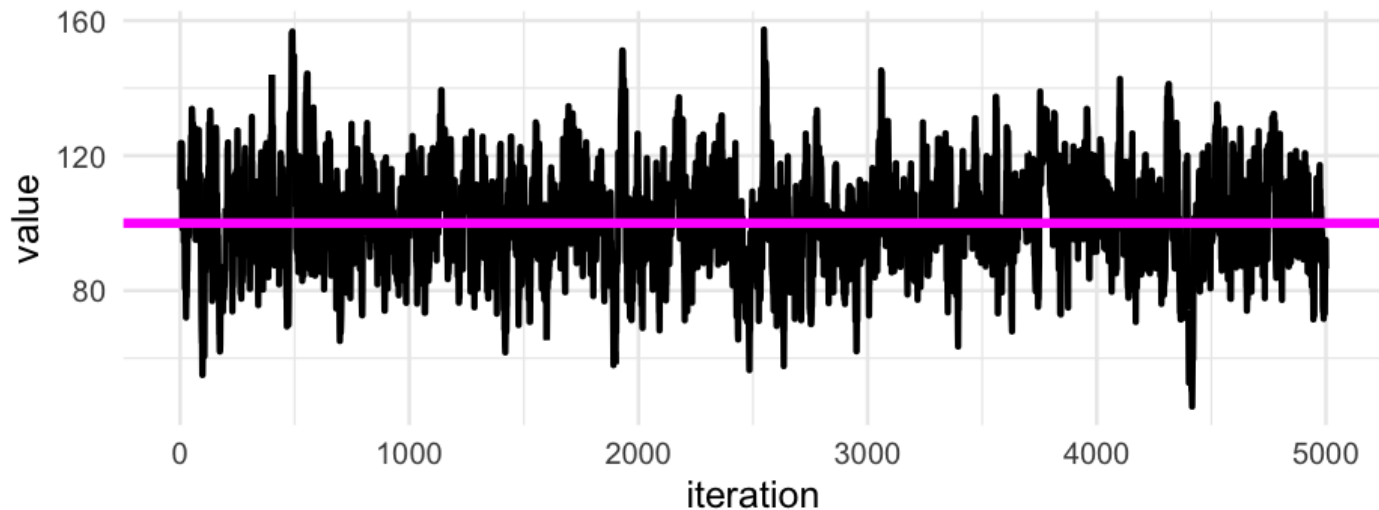
Loop

```
for(i in 2:5000) {  
  # generate proposal distribution  
  proposal <- rnorm(1, mean = samples[i - 1], sd = 10)  
  
  # calculate current/proposal distribution likelihood  
  prob_current <- dnorm(samples[i - 1], 100, 15)  
  prob_proposal <- dnorm(proposal, 100, 15)  
  
  # compute the probability ratio  
  prob_ratio <- prob_proposal / prob_current  
  
  # Determine which to select  
  if(prob_ratio > runif(1)) {  
    samples[i] <- proposal # accept  
  } else {  
    samples[i] <- samples[i - 1] # reject  
  }  
}
```

Plot

Iteration history

```
tibble(iteration = 1:5000,  
       value = samples) %>%  
  ggplot(aes(iteration, value)) +  
  geom_line() +  
  geom_hline(yintercept = 100,  
            color = "magenta",  
            size = 3)
```



Plot

Density

```
tibble(iteration = 1:5000,  
       value = samples) %>%  
  ggplot(aes(value)) +  
  geom_density(fill = "#00b4f5",  
              alpha = 0.7) +  
  geom_vline(xintercept = 100,  
            color = "magenta",  
            size = 3)
```


Bayes Update Script

Last week I promised a script to let you play around with priors. I'll show that now ("bayes-update-plotting.R")

There's also a MH script I'll quickly show.

[demo]

Implementation with {brms}

Luckily, we don't have to program the MCMC algorithm ourselves

What is it?

- **b**ayesian **r**egression **m**odeling with **s**tan
- Uses stan as the model backend – basically writes the model code for you then sends it to stan
- Allows model syntax similar to **lme4**
- Simple specification of priors – defaults are flat
- Provides many methods for post-model fitting inference

Fit a basic model

Let's start with the default (uninformative) priors, and fit a standard, simple-linear regression model

```
library(brms)
sleep_m0 <- brm(Reaction ~ Days, data = lme4::sleepstudy)
```

```
##
## SAMPLING FOR MODEL 'a3c55247ada09ee979052a314fc29ad7' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would t
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:      1 / 2000 [  0%]    (Warmup)
## Chain 1: Iteration:    200 / 2000 [ 10%]    (Warmup)
## Chain 1: Iteration:    400 / 2000 [ 20%]    (Warmup)
## Chain 1: Iteration:    600 / 2000 [ 30%]    (Warmup)
## Chain 1: Iteration:    800 / 2000 [ 40%]    (Warmup)
## Chain 1: Iteration:   1000 / 2000 [ 50%]    (Warmup)
## Chain 1: Iteration:   1001 / 2000 [ 50%]    (Sampling)
## Chain 1: Iteration:   1200 / 2000 [ 60%]    (Sampling)
## Chain 1: Iteration:   1400 / 2000 [ 70%]    (Sampling)
## Chain 1: Iteration:   1600 / 2000 [ 80%]    (Sampling)
```

Model summary

```
summary(sleep_m0)
```

```
## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: Reaction ~ Days
## Data: lme4::sleepstudy (Number of observations: 180)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##           total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    251.49      6.64   238.26   264.74 1.00     4168     3040
## Days          10.45      1.25     7.98    12.90 1.00     4155     2967
##
## Family Specific Parameters:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma        47.96      2.51    43.34    53.22 1.00     4069     2959
##
## Samples were drawn 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).
```

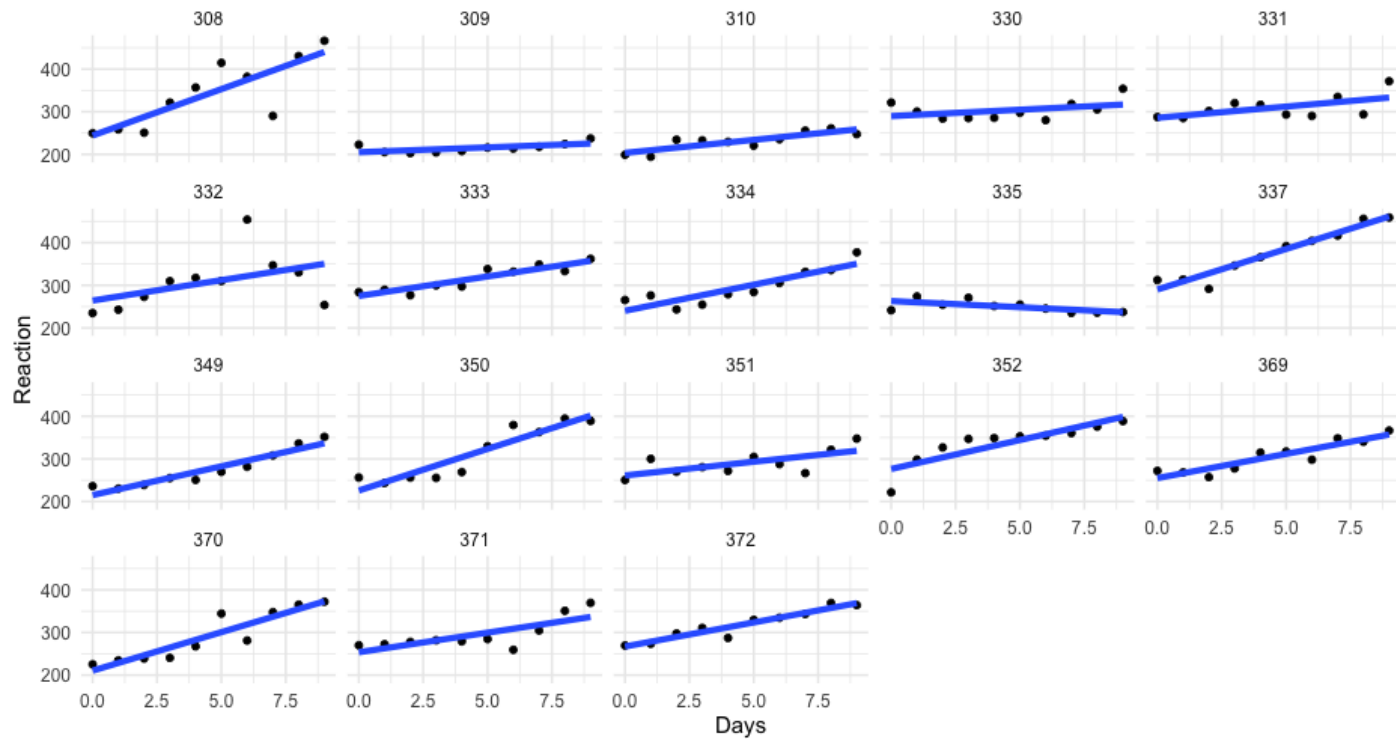
View "fixed" effect

Let's look at our estimated relation between **Days** and **Reaction**

```
conditional_effects(sleep_m0)
```

Wrong model

Of course, this is the wrong model, we have a multilevel structure



Multilevel model

Notice the syntax is essentially equivalent to **lme4**

```
sleep_m1 <- brm(Reaction ~ Days + (Days | Subject), data = lme4:
summary(sleep_m1)
```

```
## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: Reaction ~ Days + (Days | Subject)
## Data: lme4::sleepstudy (Number of observations: 180)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##           total post-warmup samples = 4000
##
## Group-Level Effects:
## ~Subject (Number of levels: 18)
##
```

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS
sd(Intercept)	27.06	7.15	15.59	42.69	1.00	1529
sd(Days)	6.55	1.49	4.19	10.04	1.00	1144
cor(Intercept, Days)	0.08	0.29	-0.47	0.64	1.01	1125

```
##
```

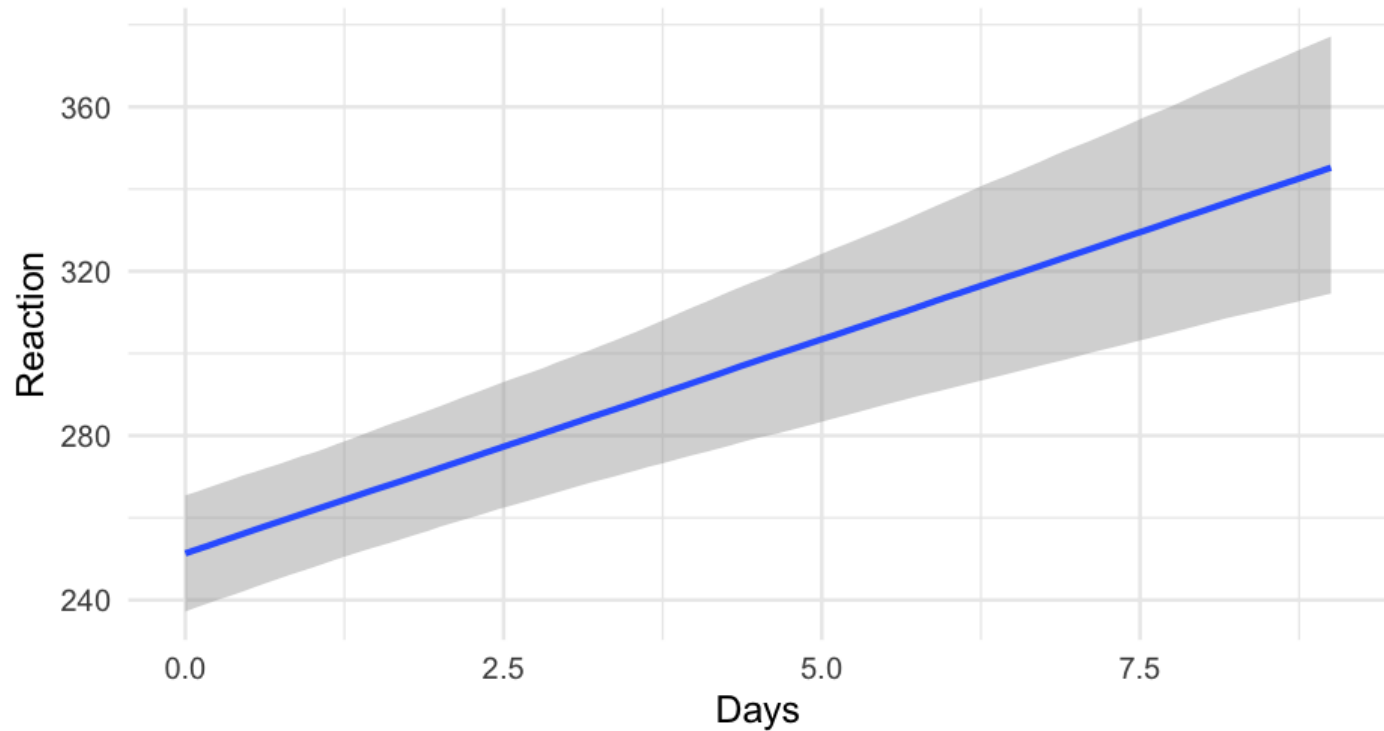
	Tail_ESS
sd(Intercept)	2353
sd(Days)	1440
cor(Intercept, Days)	1844

```
##
## Population-Level Effects:
```


Fixed effect

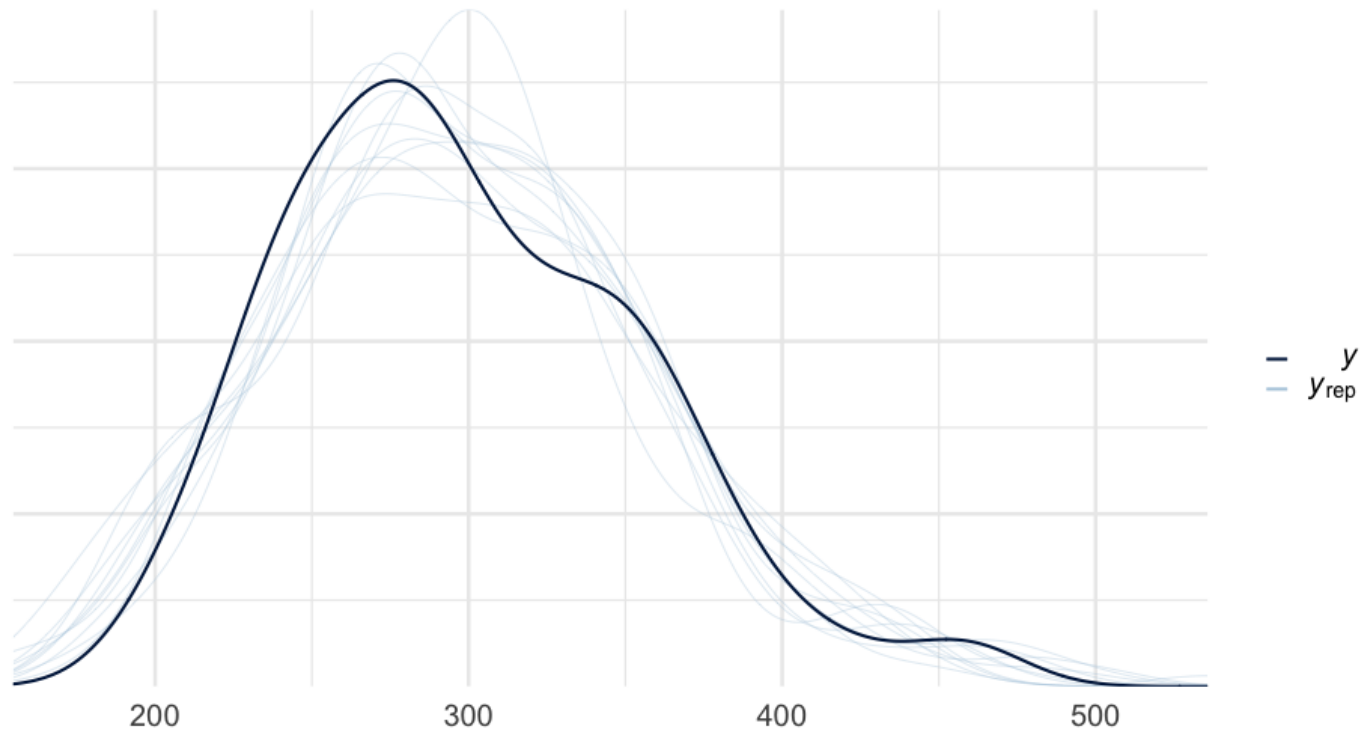
The uncertainty has increased

```
conditional_effects(sleep_m1)
```



Checking your model

```
pp_check(sleep_m1)
```



More checks

```
plot(sleep_m1)
```

Even more

```
launch_shinytan(sleep_m1)
```

Model comparison

- Two primary (modern) methods
 - Leave-one-out Cross-Validation (LOO)
 - Widely Applicable Information Criterion (WAIC)

Both provide estimates of the *out-of-sample* predictive accuracy of the model. LOO is similar to K-fold CV, while WAIC is similar to AIC/BIC (but an improved version for Bayes models)

Both can be computed using **brms**. LOO is approximated (not actually refit each time).

LOO: M0

```
loo(sleep_m0)
```

```
##
## Computed from 4000 by 180 log-likelihood matrix
##
##           Estimate      SE
## elpd_loo    -953.3  10.5
## p_loo         3.2   0.5
## looic       1906.5  21.0
## -----
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

LOO: M1

```
loo(sleep_m1)
```

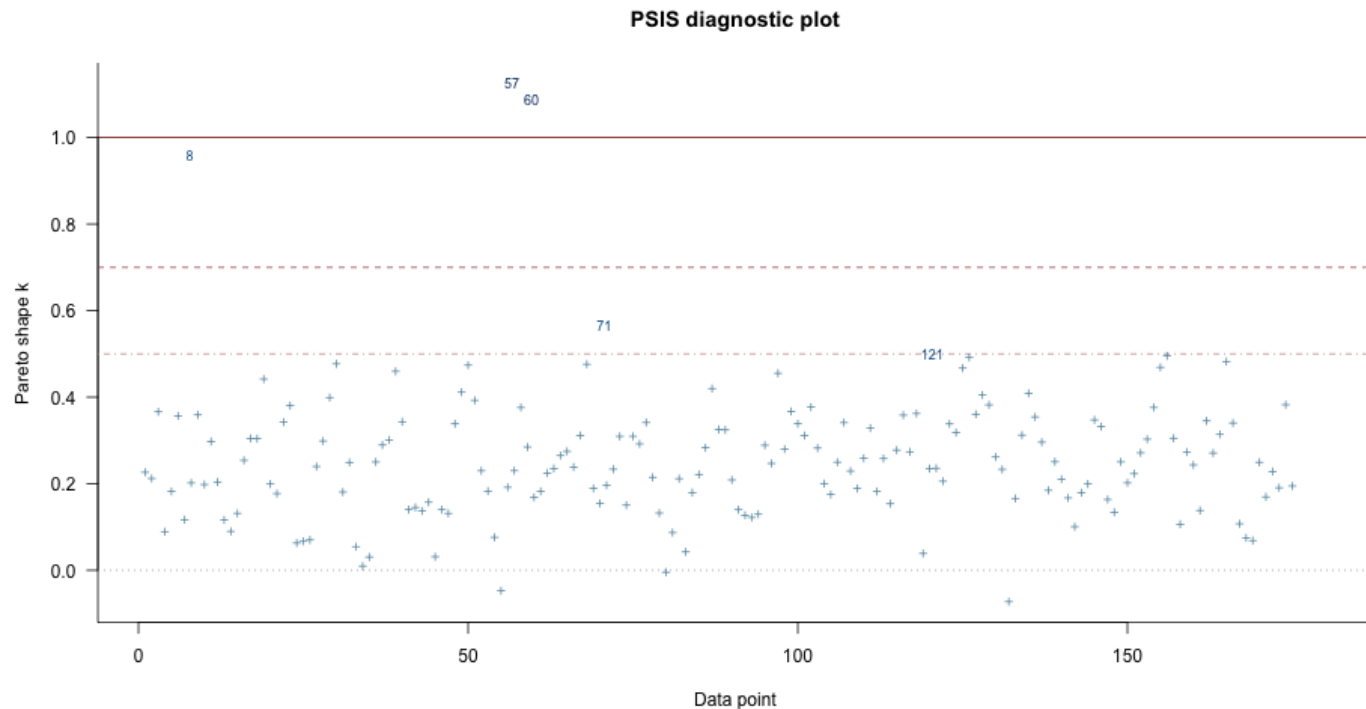
```
## Warning: Found 3 observations with a pareto_k > 0.7 in model 'sleep_m1'.  
## It is recommended to set 'moment_match = TRUE' in order to perform moment  
## matching for problematic observations.
```

```
##  
## Computed from 4000 by 180 log-likelihood matrix  
##  
##           Estimate      SE  
## elpd_loo    -861.7  22.7  
## p_loo        34.7   8.8  
## looic       1723.3 45.5  
## -----  
## Monte Carlo SE of elpd_loo is NA.  
##  
## Pareto k diagnostic values:  
##           Count Pct.      Min. n_eff  
## (-Inf, 0.5]   (good)    175    97.2%    682  
##  (0.5, 0.7]   (ok)       2     1.1%   2189  
##   (0.7, 1]    (bad)       1     0.6%    14  
##   (1, Inf)    (very bad)  2     1.1%     9  
## See help('pareto-k-diagnostic') for details.
```

Plot

```
plot(loo(sleep_m1), label_points = TRUE)
```

```
## Warning: Found 3 observations with a pareto_k > 0.7 in model 'sleep_m1'.  
## It is recommended to set 'moment_match = TRUE' in order to perform momen  
## matching for problematic observations.
```



Interpretation

Large values here (anything above 0.7) is an indication of model misspecification

Smaller values do not guarantee a well-specified model, however

LOO Compare

The best fitting model will be on top. Use the standard error within your interpretation

```
loo_compare(loo(sleep_m0), loo(sleep_m1))
```

```
## Warning: Found 3 observations with a pareto_k > 0.7 in model 'sleep_m1'.  
## It is recommended to set 'moment_match = TRUE' in order to perform moment  
## matching for problematic observations.
```

```
##           elpd_diff se_diff  
## sleep_m1      0.0      0.0  
## sleep_m0    -91.6     21.4
```

This is a fairly complicated topic, and we won't spend a lot of time on it. See [here](#) for a bit more information specifically on the functions

WAIC

Similar to other information criteria.

```
waic(sleep_m0)
```

```
##
## Computed from 4000 by 180 log-likelihood matrix
##
##           Estimate      SE
## elpd_waic    -953.3  10.5
## p_waic         3.2   0.5
## waic          1906.5 21.0
```

```
waic(sleep_m1)
```

```
## Warning:
## 14 (7.8%) p_waic estimates greater than 0.4. We recommend trying loo ins

##
## Computed from 4000 by 180 log-likelihood matrix
##
##           Estimate      SE
## elpd_waic    -860.2  22.3
## p_waic         33.2   8.4
```

Compare

You can use `waic` within `loo_compare()`

```
loo_compare(waic(sleep_m0), waic(sleep_m1))
```

```
## Warning:
```

```
## 14 (7.8%) p_waic estimates greater than 0.4. We recommend trying loo ins
```

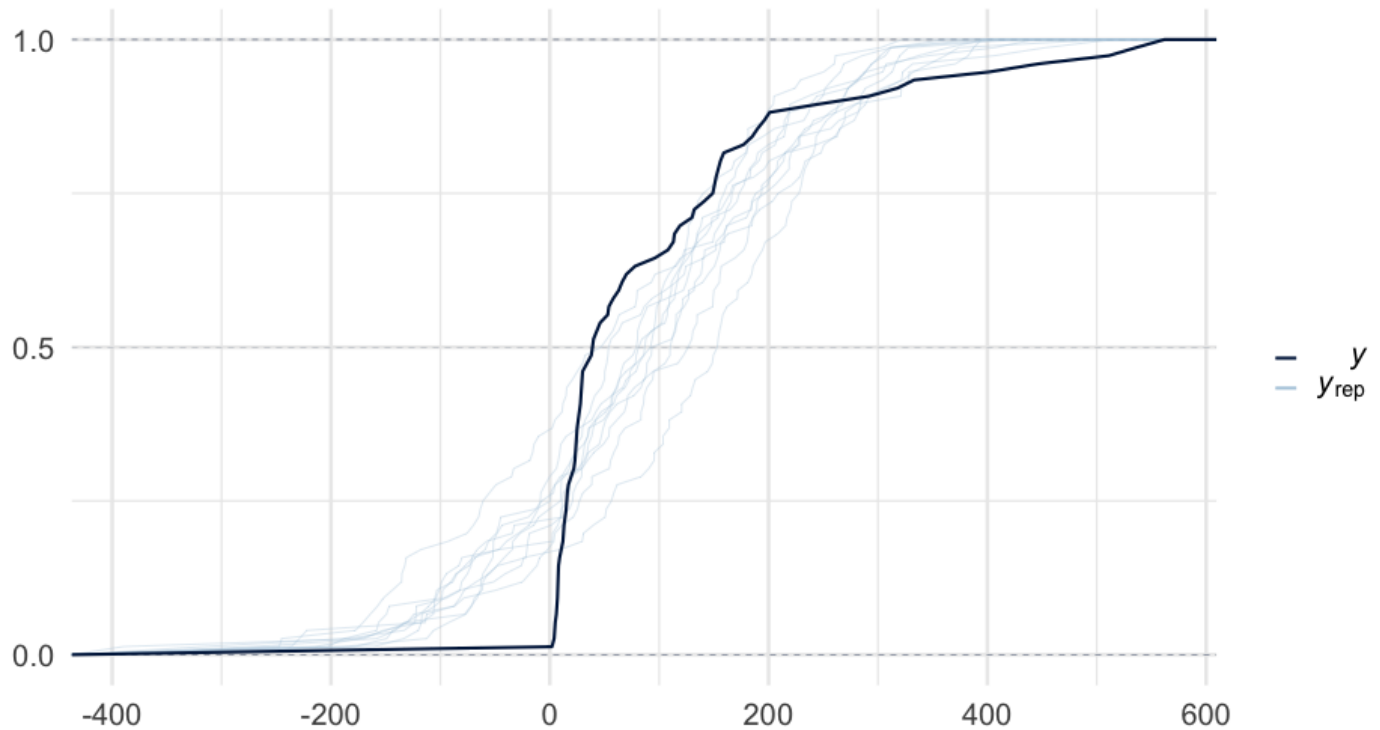
```
##           elpd_diff se_diff
```

```
## sleep_m1      0.0      0.0
```

```
## sleep_m0    -93.0     21.0
```

Another model

```
kidney_m0 <- brm(time ~ age + sex, data = kidney)  
pp_check(kidney_m0, type = "ecdf_overlay")
```



Fixing this

We need to change the assumptions of our model – specifically that the outcome is not normally distributed

Plot the raw data

```
ggplot(kidney, aes(time)) +  
  geom_histogram(alpha = 0.7)
```

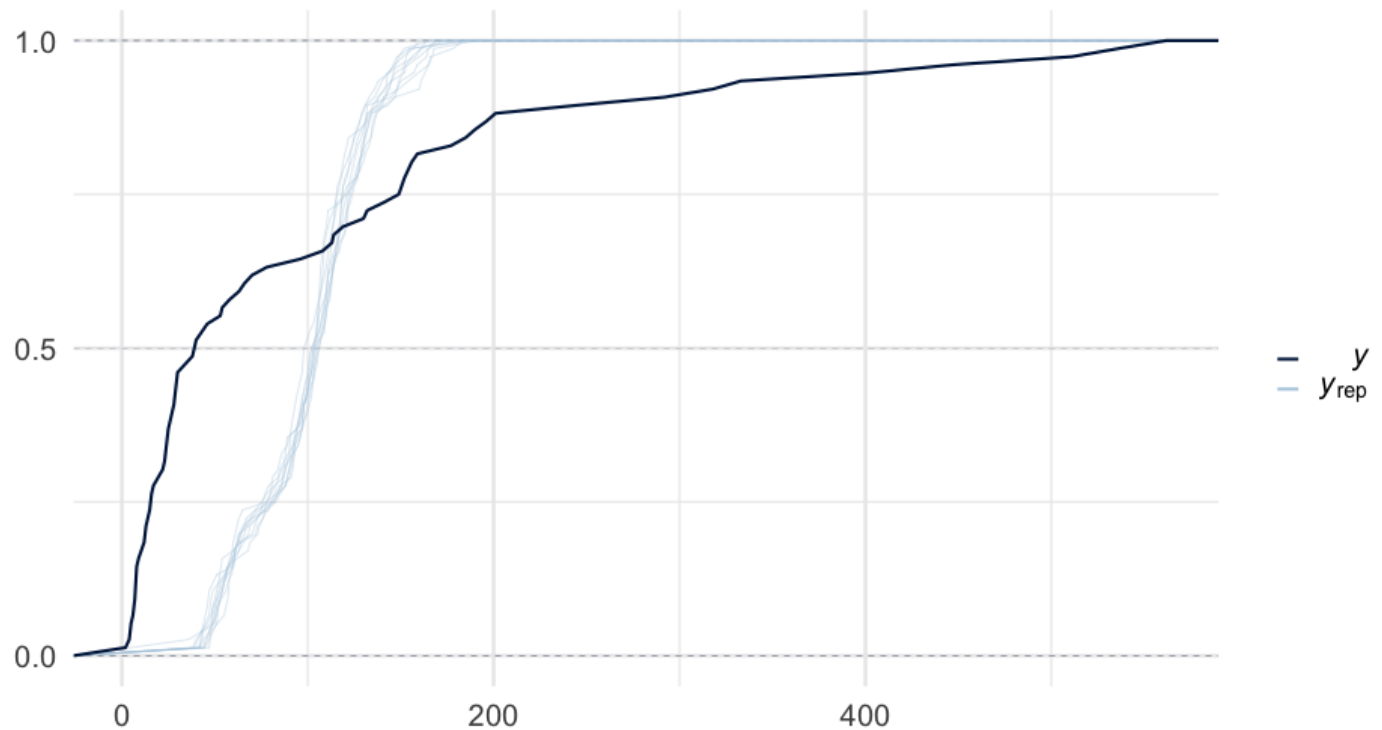
Maybe Poisson?

(I realize we haven't talked about these types of models yet)

```
kidney_m1 <- brm(time ~ age + sex,  
                 data = kidney,  
                 family = poisson())
```

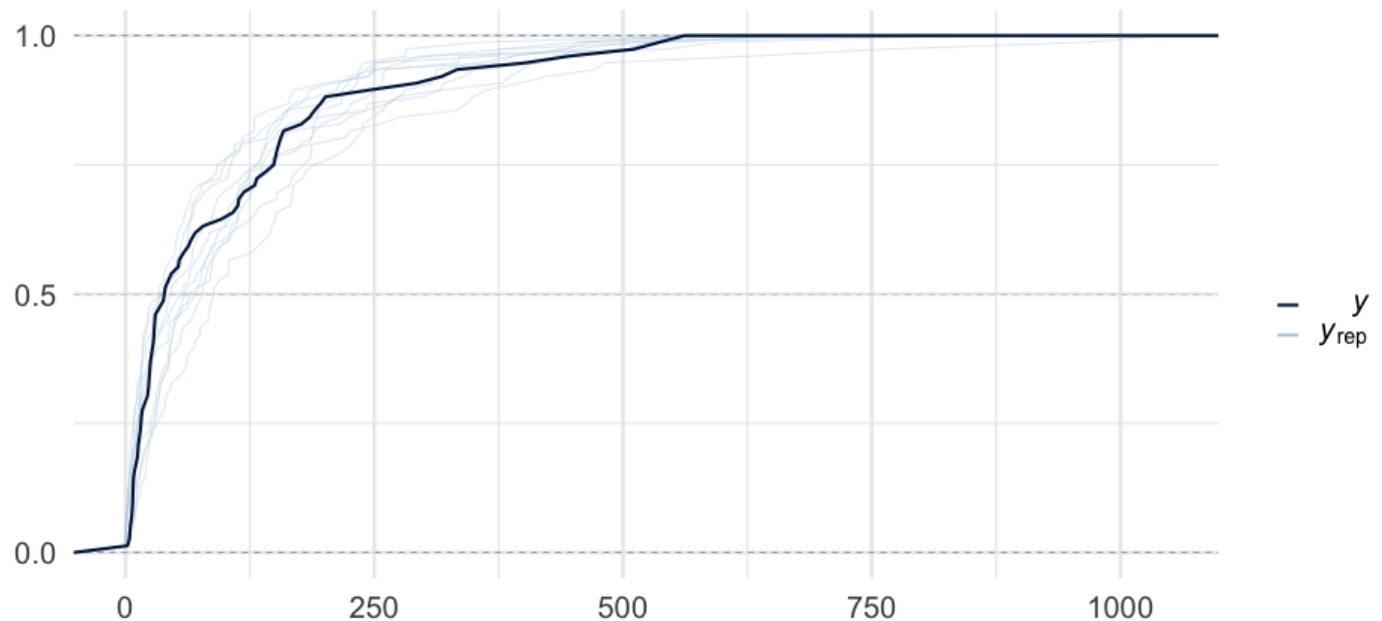
Nope

```
pp_check(kidney_m1, type = "ecdf_overlay")
```



Gamma w/log link

```
kidney_m2 <- brm(time ~ age + sex,  
                  data = kidney,  
                  family = Gamma("log"))  
pp_check(kidney_m2, type = "ecdf_overlay")
```



Specifying priors

Let's sample from *only* our priors to see what kind of predictions we get.

Here, we're specifying that our beta coefficient prior is $\beta \sim N(0, 0.5)$

```
kidney_m3 <- brm(  
  time ~ age + sex,  
  data = kidney,  
  family = Gamma("log"),  
  prior = prior(normal(0, 0.5), class = "b"),  
  sample_prior = "only"  
)
```

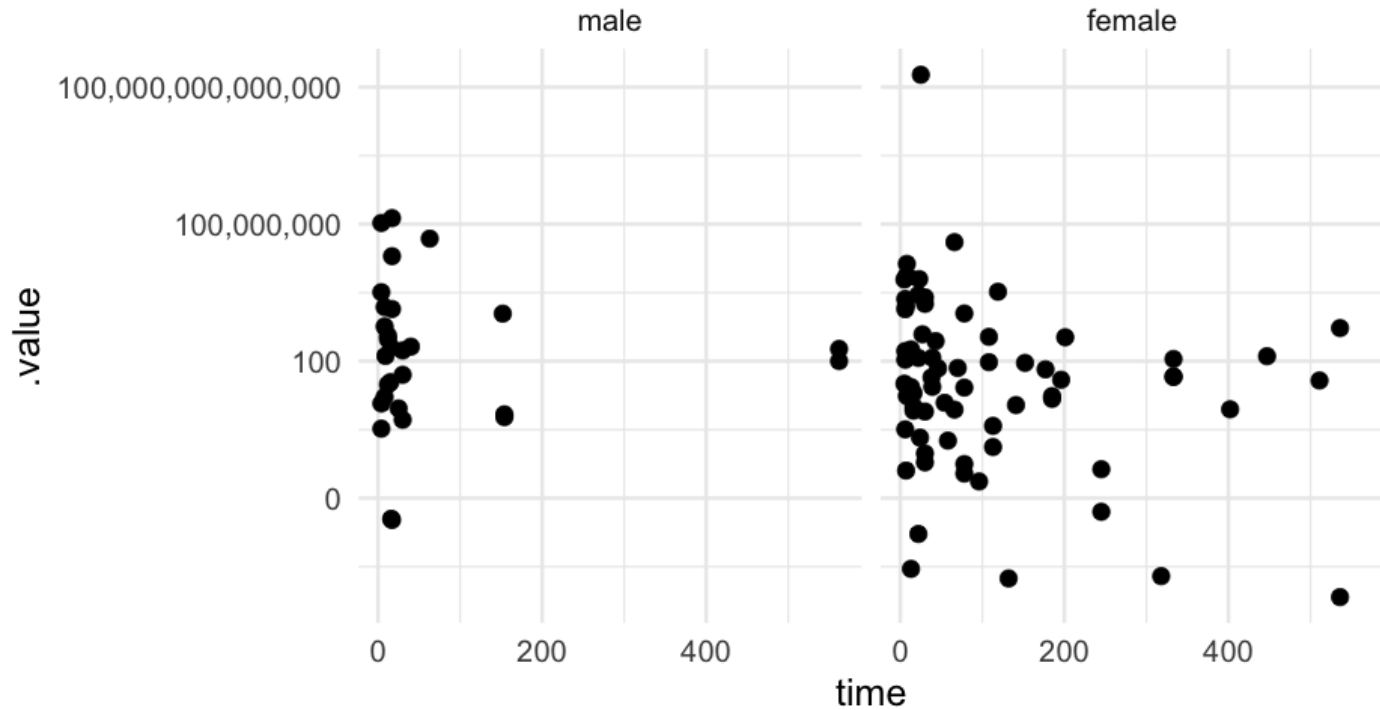
kidney_m3

```
## Warning: There were 2253 divergent transitions after warmup.
## Increasing adapt_delta above 0.8 may help. See http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Family: gamma
## Links: mu = log; shape = identity
## Formula: time ~ age + sex
## Data: kidney (Number of observations: 76)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##           total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      4.23      22.42  -39.94   46.55 1.01      629      991
## age            -0.02       0.51   -0.97    0.97 1.01      629      946
## sexfemale       0.01       0.50   -0.98    0.97 1.01      667     1196
##
## Family Specific Parameters:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## shape          0.42       3.64    0.00    2.35 1.01      436      711
##
## Samples were drawn 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).
```

Prior predictions

Random sample of 100 points



Why?

It seemed like our prior was fairly tight

The exploding prior happens because of the log transformation

- Age is coded in years
- Imagine a coef of 1 (2 standard deviations above our prior)
- Prediction for a 25 year old would be **$\exp(25) = 7.2004899 \times 10^{10}$**

A note on prior specifications

- It's hard
- I don't have a ton of good advice
- Be particularly careful when you're using distributions that have anything other than an identity link (e.g., log link, as we are here)

One more model

Let's fit a model we've fit previously

In Week 4, we fit this model

```
library(lme4)
popular <- read_csv(here::here("data", "popularity.csv"))
m_lmer <- lmer(popular ~ extrav + (extrav|class), popular,
               control = lmerControl(optimizer = "bobyqa"))
```

Try fitting the same model with **{brms}** with the default, diffuse priors

02:00

Bayesian version

```
m_brms <- brm(popular ~ extrav + (extrav|class), popular)
```

```
## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: popular ~ extrav + (extrav | class)
## Data: popular (Number of observations: 2000)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##           total post-warmup samples = 4000
##
## Group-Level Effects:
## ~class (Number of levels: 100)
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)      1.72      0.17    1.41    2.08 1.00    1165
## sd(extrav)         0.16      0.03    0.11    0.22 1.01     724
## cor(Intercept,extrav) -0.95      0.03   -1.00   -0.89 1.01    358
##           Tail_ESS
## sd(Intercept)    2052
## sd(extrav)       1755
## cor(Intercept,extrav) 564
##
## Population-Level Effects:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      2.46      0.20    2.06    2.84 1.00     994    1951
## extrav         0.49      0.03    0.44    0.54 1.00    1510    2474
##
## Family Specific Parameters:
```


lme4 model

```
arm::display(m_lmer)
```

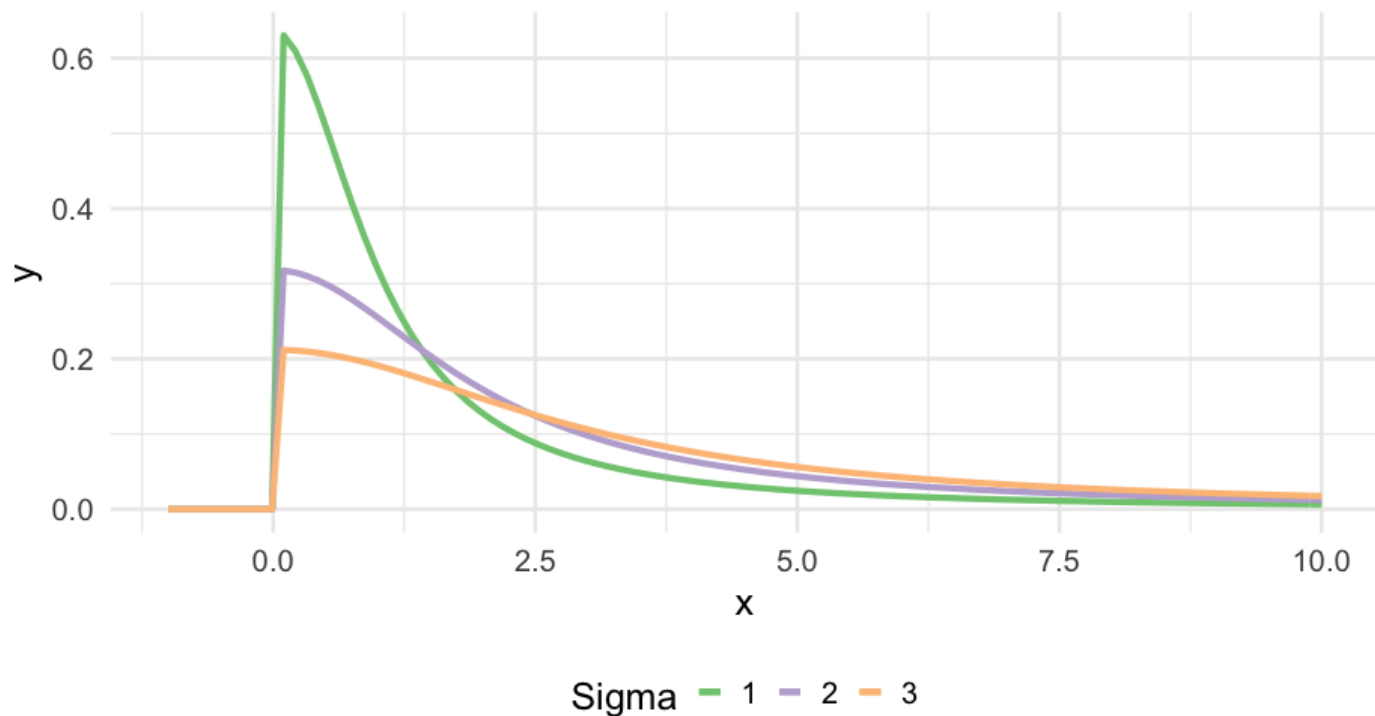
```
## lmer(formula = popular ~ extrav + (extrav | class), data = popular,  
##       control = lmerControl(optimizer = "bobyqa"))  
##               coef.est coef.se  
## (Intercept)  2.46      0.20  
## extrav      0.49      0.03  
##  
## Error terms:  
##   Groups   Name              Std.Dev.  Corr  
##   class    (Intercept)  1.73  
##           extrav        0.16      -0.97  
## Residual                    0.95  
## ---  
## number of obs: 2000, groups: class, 100  
## AIC = 5791.4, DIC = 5762  
## deviance = 5770.7
```

Add priors

- Multiple ways to do this
- Generally, I allow the intercept to just be what it is
- For "fixed" effects, you might consider regularizing priors.
- You can also set priors for individual parameters
- In my experience, you will mostly want to set priors for different "classes" of parameters, e.g.: **Intercept**, **b**, **sd**, **sigma**, **cor**

Half-Cauchy distribution

This is the distribution most often used for standard deviation priors



Specify some new priors

Let's specify some regularizing priors for the fixed effects and standard deviations

```
priors <- c(
  prior(normal(0, 0.5), class = b),
  prior(cauchy(0, 1), class = sd)
)

m_brms2 <- brm(popular ~ extrav + (extrav|class),
  data = popular,
  prior = priors)
```

Almost no difference in this case

m_brms2

```
## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: popular ~ extrav + (extrav | class)
## Data: popular (Number of observations: 2000)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##           total post-warmup samples = 4000
##
## Group-Level Effects:
## ~class (Number of levels: 100)
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)      1.70      0.17    1.39    2.06 1.01    1358
## sd(extrav)         0.16      0.03    0.11    0.22 1.00     871
## cor(Intercept,extrav) -0.95      0.03   -1.00   -0.88 1.01    437
##           Tail_ESS
## sd(Intercept)      1824
## sd(extrav)         1683
## cor(Intercept,extrav) 871
##
## Population-Level Effects:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      2.48      0.20    2.08    2.86 1.00    1157    1810
## extrav         0.49      0.03    0.44    0.54 1.00    1960    2642
##
## Family Specific Parameters:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
```

Speed

We'll never reach the `lme4::lmer()` speeds, but we can make it faster.

- Parallelize (not always as effective as you might hope)
- Use the cmdstanr backend
 - This requires a little bit of additional work, but is probably worth it if you're fitting bigger models.

```
install.packages(  
  "cmdstanr",  
  repos = c(  
    "https://mc-stan.org/r-packages/",  
    getOption("repos")  
  )  
)
```

Timings

I'm not evaluating the below, but the timings were 114.368 seconds and 87.383 seconds, respectively.

```
library(tictoc)

tic()
m_brms <- brm(popular ~ extrav + (extrav|class),
              data = popular)
toc()

tic()
m_brms2 <- brm(popular ~ extrav + (extrav|class),
               data = popular,
               backend = "cmdstanr")
toc()
```

Wrapping up
our Bayes
intro

Advantages to Bayes

- Opportunity to incorporate prior knowledge into the modeling process (you don't really *have* to – could just set wide priors)
- Natural interpretation of uncertainty
- Can often allow you to estimate models that are difficult if not impossible with frequentist methods

Disadvantages

- Generally going to be slower in implementation
- You may run into pushback from others – particularly with respect to priors

Notes on the posterior

- The posterior is the distribution of the parameters, given the data
- Think of it as the distribution of what we don't know, but are interested in (model parameters), given what we know or have observed (the data), and our prior beliefs
- Gives a complete picture of parameter uncertainty
- We can do lots of things with the posterior that is hard to get otherwise

Break

05:00

Review of logistic regression

Data generating distribution

Up to this point, we've been assuming the data were generated from a normal distribution.

For example, we might fit a simple linear regression model to the wages data like this

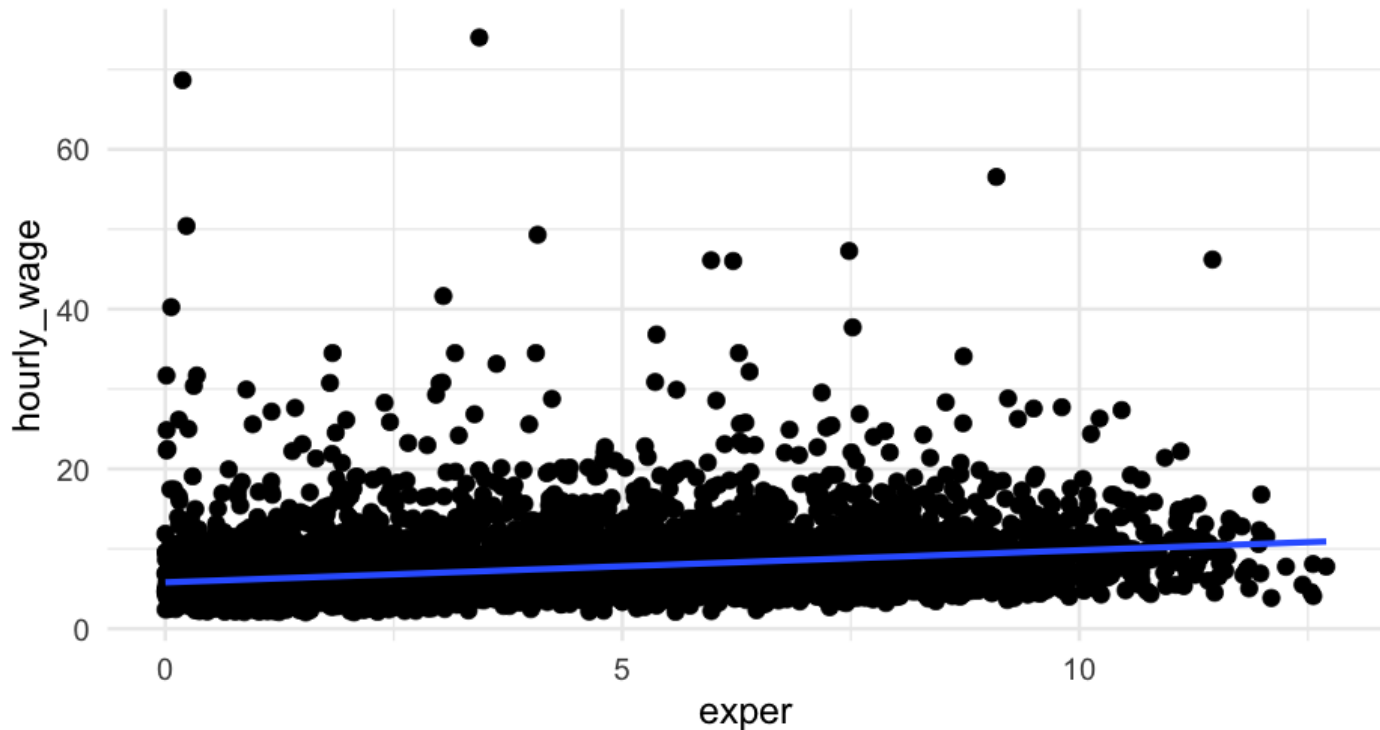
$$\text{wages}_i \sim N(\hat{y}, \sigma^2)$$
$$\hat{y} = \alpha + \beta_1(\text{exper})$$

where we're embedding a linear model into the mean in code

```
wages <- read_csv(here::here("data", "wages.csv")) %>%  
  mutate(hourly_wage = exp(lnw))  
  
wages_lm <- lm(hourly_wage ~ exper, data = wages)
```

Graphically

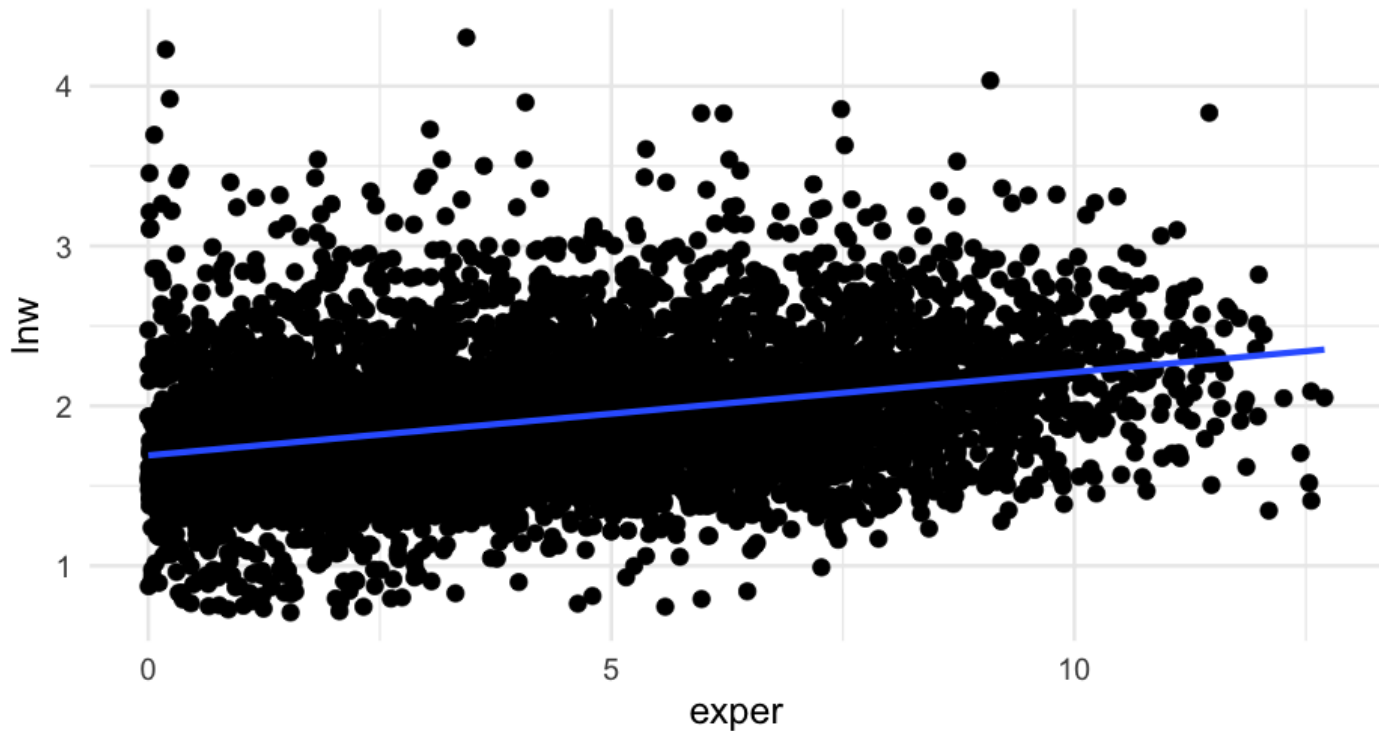
```
ggplot(wages, aes(exper, hourly_wage)) +  
  geom_point() +  
  geom_smooth(method = "lm", se = FALSE)
```



Aside – you see why the log of wages was modeled instead?

Log wages

```
ggplot(wages, aes(exper, lnw)) +  
  geom_point() +  
  geom_smooth(method = "lm", se = FALSE)
```



Move to binary model

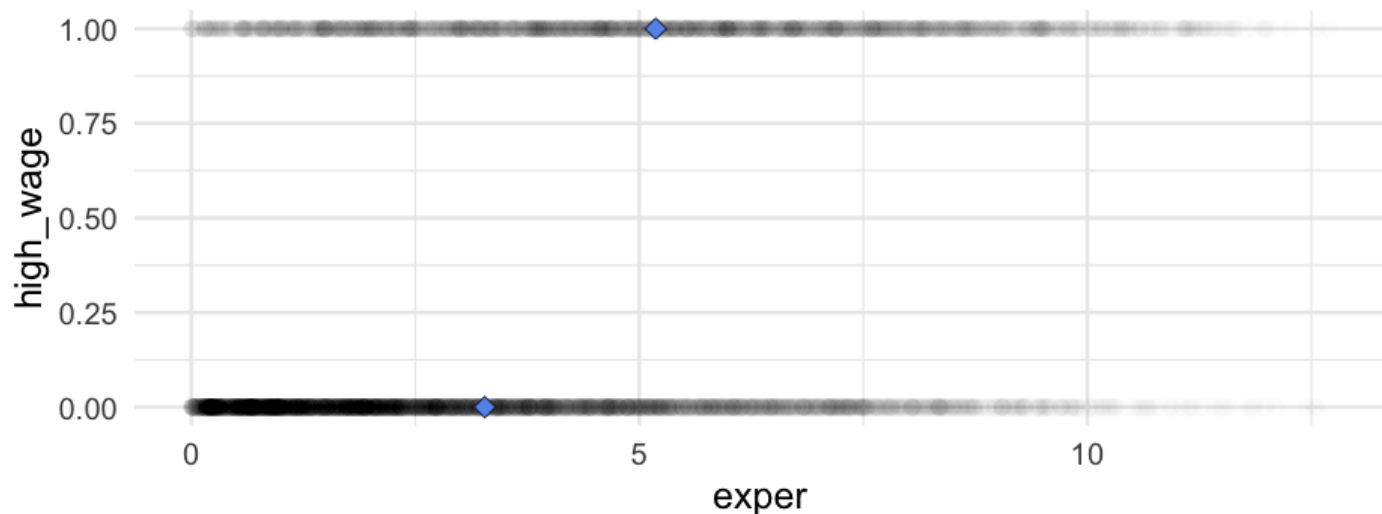
Let's split the wages data into a binary classification based on whether it's above or below the mean.

```
wages <- wages %>%  
  mutate(  
    high_wage = ifelse(  
      hourly_wage > mean(hourly_wage, na.rm = TRUE), 1, 0  
    )  
  )  
  
wages %>%  
  select(id, hourly_wage, high_wage)
```

```
## # A tibble: 6,402 x 3  
##       id hourly_wage high_wage  
##   <dbl>      <dbl>    <dbl>  
## 1     31    4.441535        0  
## 2     31    4.191254        0  
## 3     31    4.344888        0  
## 4     31    5.748851        0  
## 5     31    6.896403        0  
## 6     31    5.523435        0  
## 7     31    8.052640        1
```


Plot

```
means <- wages %>%  
  group_by(high_wage) %>%  
  summarize(mean = mean(exper))  
  
ggplot(wages, aes(exper, high_wage)) +  
  geom_point(alpha = 0.01) +  
  geom_point(aes(x = mean), data = means,  
              shape = 23,  
              fill = "cornflowerblue")
```



We could fit a linear model

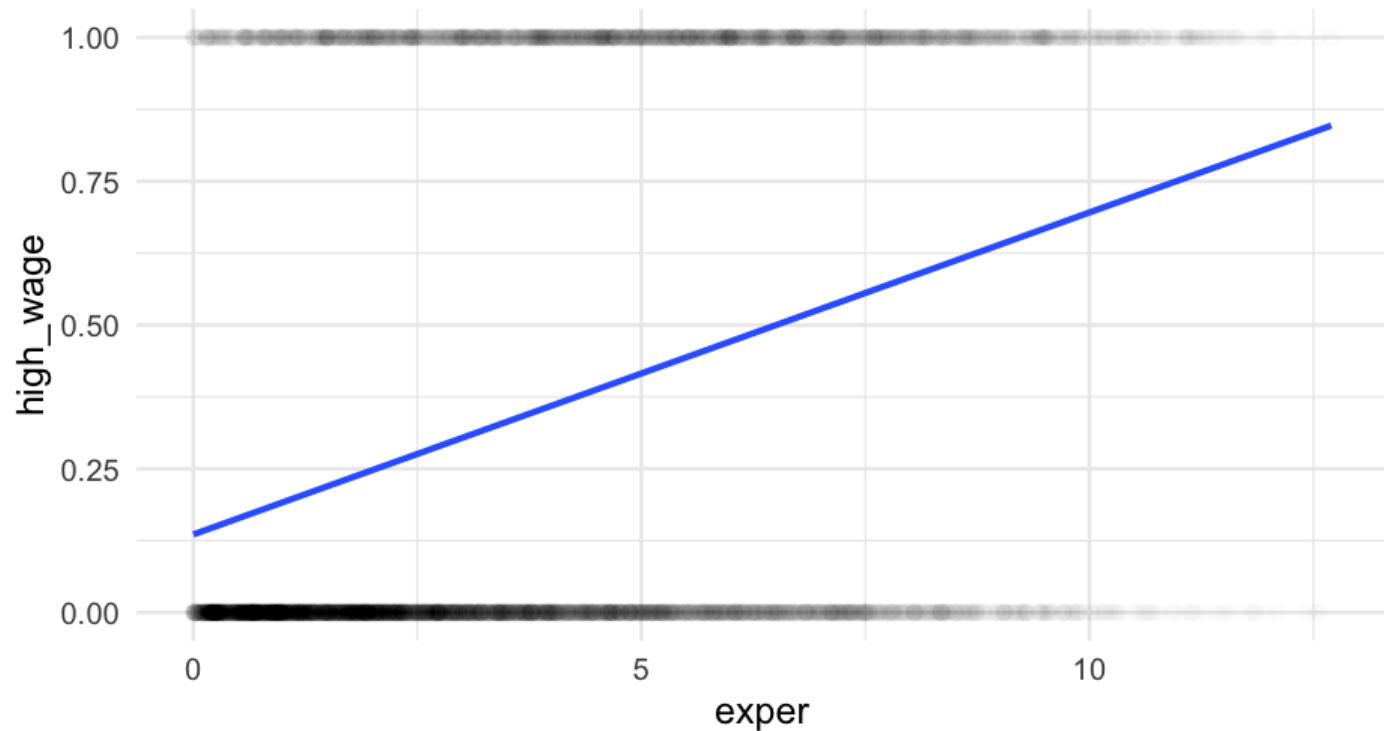
```
m_lpm <- lm(high_wage ~ exper, data = wages)
arm::display(m_lpm)
```

```
## lm(formula = high_wage ~ exper, data = wages)
##               coef.est coef.se
## (Intercept)  0.14      0.01
## exper        0.06      0.00
## ---
## n = 6402, k = 2
## residual sd = 0.45, R-Squared = 0.11
```

This is referred to as a linear probability model (LPM) and they are pretty hotly contested, with proponents and detractors

Plot

```
ggplot(wages, aes(exper, high_wage)) +  
  geom_point(alpha = 0.01) +  
  geom_smooth(method = "lm", se = FALSE)
```



Prediction

What if somebody has an experience of 25 years?

```
predict(m_lpm, newdata = data.frame(exper = 25))
```

```
##          1  
## 1.535723
```



Our prediction goes outside the range of our data.

As a rule, the assumed data generating process should match the boundaries of the data.

Of course, you could truncate after the fact. I think that's less than ideal (see [here](#) or [here](#) for discussions contrasting LPM with logistic regression).

The binomial model

$$y_i \sim \text{Binomial}(n, p_i)$$

Think in terms of coin flips

n is the number of coin flips, while p_i is the probability of heads

Example

Flip 1 coin 10 times

```
set.seed(42)
rbinom(
  n = 10, # number of trials
  size = 1, # number of coins
  prob = 0.5 # probability of heads
)
```

```
## [1] 1 1 0 1 1 1 1 0 1 1
```

Side note – a binomial model with `size = 1` (or $n = 1$ in equation form) is equivalent to a Bernoulli distribution

Flip 10 coins 1 time

```
rbinom(n = 1, size = 10, prob = 0.5)
```

```
## [1] 5
```

Modeling

We now build a linear model for \mathbf{p} , just like we previously built a linear model for μ .

A problem

Probability is bounded $[0, 1]$

We need to ensure that our model respects these bounds

Link functions

We solve this problem by using a *link* function

$$y_i \sim \text{Binomial}(n, p_i)$$
$$f(p_i) = \alpha + \beta(x_i)$$

- Instead of modeling p_i directly, we model $f(p_i)$
- The specific f is the link function
- Link functions map the *linear* space of the model to the *non-linear* parameters (like probability)
- The *log* and *logit* links are most common

Binomial logistic regression

If we only have two categories, we commonly assume a binomial distribution, with a logit link.

$$y_i \sim \text{Binomial}(n, p_i)$$
$$\text{logit}(p_i) = \alpha + \beta(x_i)$$

where the logit link is defined by the *log-odds*

$$\text{logit}(p_i) = \log \left[\frac{p_i}{1 - p_i} \right]$$

So

$$\log \left[\frac{p_i}{1 - p_i} \right] = \alpha + \beta(x_i)$$

Inverse link

What we probably want to interpret is probability

We can transform the log-odds to probability by exponentiating

$$p_i = \frac{\exp(\alpha + \beta(x_i))}{1 + \exp(\alpha + \beta(x_i))}$$

This is the logistic function, or the inverse-logit.

Example logistic regression model

```
m_glm <- glm(high_wage ~ exper,  
             data = wages,  
             family = binomial(link = "logit"))  
arm::display(m_glm)
```

```
## glm(formula = high_wage ~ exper, family = binomial(link = "logit"),  
##      data = wages)  
##               coef.est coef.se  
## (Intercept)  -1.65      0.05  
## exper         0.25      0.01  
## ---  
##      n = 6402, k = 2  
##      residual deviance = 7655.2, null deviance = 8345.8 (difference = 690.5)
```

Coefficient interpretation

The coefficients are reported on the *log-odds* scale. Other than that, interpretation is the same.

For example:

The log-odds of a participant with zero years experience being in the high wage category was -1.65 .

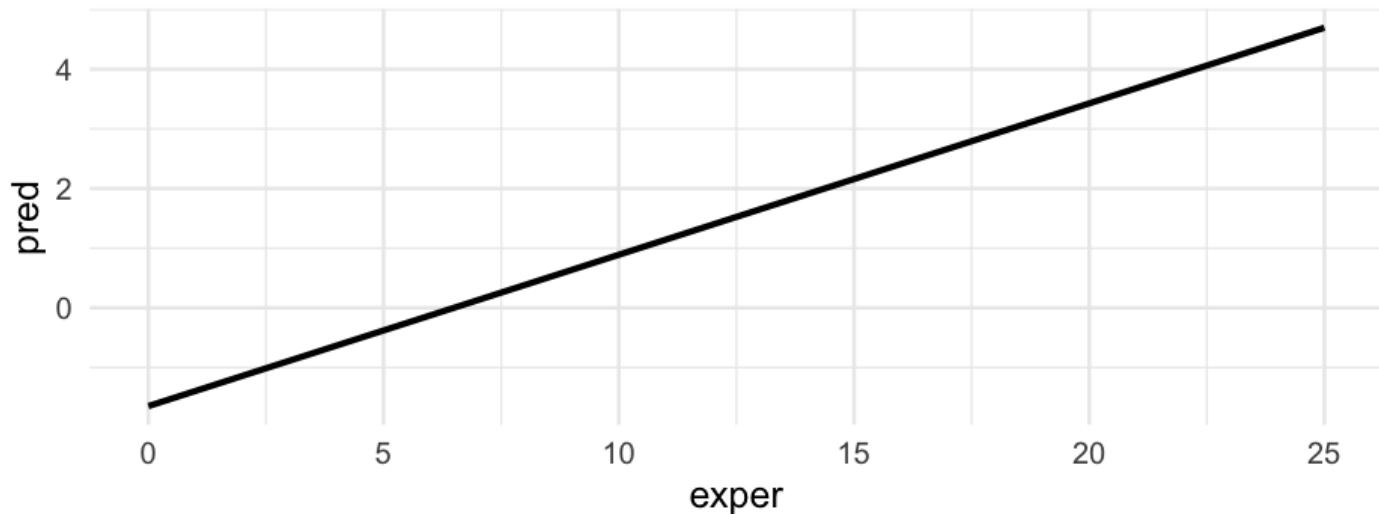
For every one year of additional experience, the log-odds of being in the high wage category increased by 0.25 .

Note

- Outside of scientific audiences, almost nobody is going to understand the previous slide
- You *cannot* just transform the coefficients and interpret them as probabilities (because it is non-linear on the probability scale).

Log odds scale

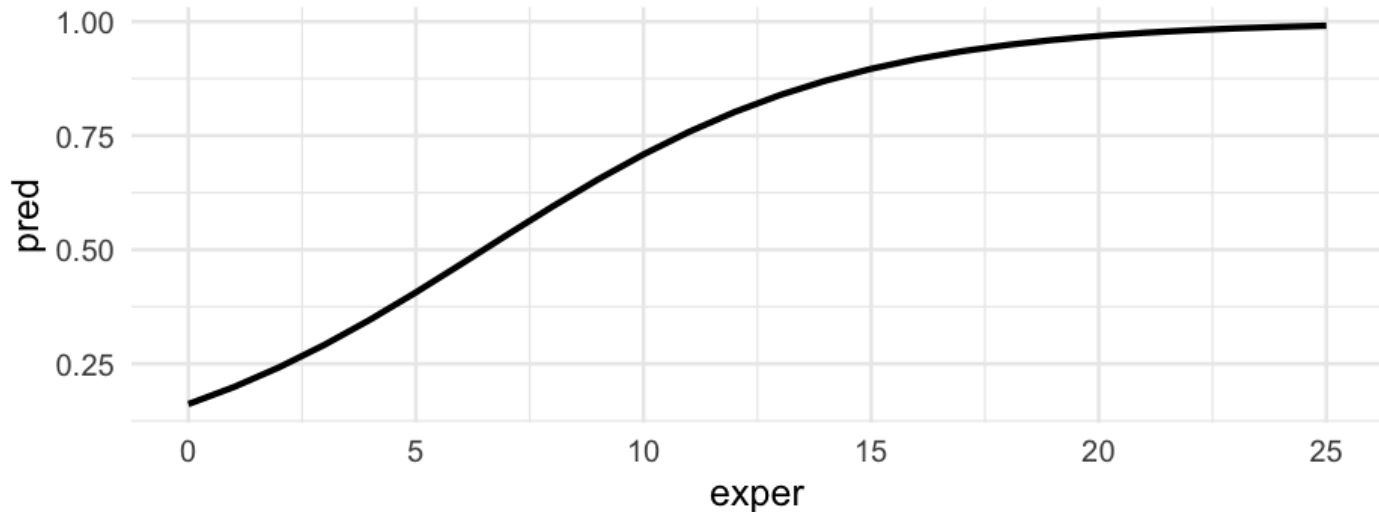
```
tibble(exper = 0:25) %>%  
  mutate(pred = predict(m_glm, newdata = .)) %>%  
  ggplot(aes(exper, pred)) +  
  geom_line()
```



Perfectly straight line – change in log-odds are modeled as a linear function of experience

Probability scale

```
tibble(exper = 0:25) %>%  
  mutate(pred = predict(m_glm,  
                        newdata = .,  
                        type = "response")) %>%  
  ggplot(aes(exper, pred)) +  
  geom_line()
```



Our model parameters map to probability non-linearly, and it is bound to $[0, 1]$

Probability predictions

Let's make the predictions from the previous slide "by hand"

$$p_i = \frac{\exp(\alpha + \beta(x_i))}{1 + \exp(\alpha + \beta(x_i))}$$

And our coefficients are

```
coef(m_glm)
```

```
## (Intercept)      exper  
## -1.6467568    0.2536815
```


Intercept

$$(p_i | \text{exper} = 0) = \frac{\exp(-1.65 + 0.25(0))}{1 + \exp(-1.65 + 0.25(0))}$$

$$(p_i | \text{exper} = 0) = \frac{\exp(-1.65)}{1 + \exp(-1.65)}$$

$$(p_i | \text{exper} = 0) = \frac{0.19}{1.19} = 0.16$$

Five years experience

Notice the exponentiation happens *after* adding the coefficients together

$$(p_i | \text{exper} = 5) = \frac{\exp(-1.65 + 0.25(5))}{1 + \exp(-1.65 + 0.25(5))}$$

$$(p_i | \text{exper} = 5) = \frac{\exp(-0.4)}{1 + \exp(-0.4)}$$

$$(p_i | \text{exper} = 5) = \frac{0.67}{1.67} = 0.40$$

Fifteen years experience

$$(p_i | \text{exper} = 15) = \frac{\exp(-1.65 + 0.25(15))}{1 + \exp(-1.65 + 0.25(15))}$$

$$(p_i | \text{exper} = 15) = \frac{\exp(2.1)}{1 + \exp(2.1)}$$

$$(p_i | \text{exper} = 15) = \frac{8.16}{9.16} = 0.89$$

More interpretation

Let's try to make this so more people might understand.

First, show the logistic curve on the probability scale! Can help prevent linear interpretations

Discuss probabilities at different x values

The "divide by 4" rule

Divide by 4

- Logistic curve is steepest when $p_i = 0.5$
- The slope of the logistic curve (its derivative) is maximized at $\beta/4$
- Aside from the intercept, we can say that the change is *no more* than $\beta/4$

Example

$$\frac{0.25}{4} = 0.0625$$

So, a one year increase in experience corresponds to, at most, a 6% increase in being in the high wage category

Example writeup

There was a 16 percent chance that a participant with zero years experience would be in the *high wage* category. The logistic function, which mapped years of experience to the probability of being in the *high-wage* category, was non-linear, as shown in Figure 1. At its steepest point, a one year increase in experience corresponded with approximately a 6% increase in the probability of being in the high-wage category. For individuals with 5, 10, and 15 years of experience, the probability increased to a 41, 71, and 90 percent chance, respectively.

Other

distributions

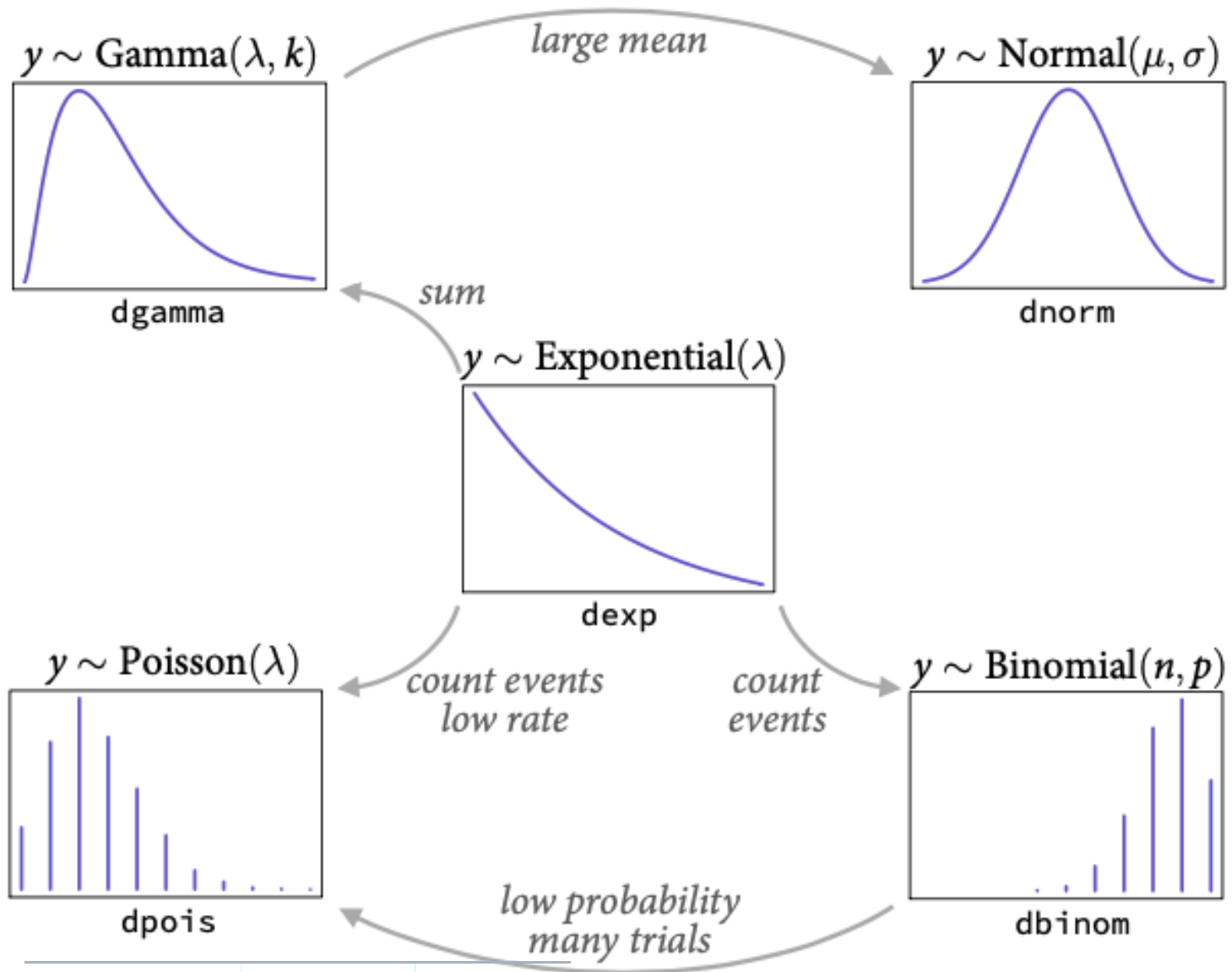


Figure from [McElreath](#)

Multilevel logistic regression

The data

Polling data from the 1988 election.

```
polls <- rio::import(here::here("data", "polls.dta"),
                     setclass = "tbl_df")
polls
```

```
## # A tibble: 13,544 x 10
##   org   year survey  bush state  edu   age female black weight
##   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1     1     1     1     7     2     2     1     0    1403
## 2     1     1     1     1    33     4     3     0     0     778
## 3     1     1     1     0    20     2     1     1     0    1564
## 4     1     1     1     1    31     3     2     1     0    1055
## 5     1     1     1     1    18     3     1     1     0    1213
## 6     1     1     1     1    31     4     2     0     0     910
## 7     1     1     1     1    40     1     3     0     0     735
## 8     1     1     1     1    33     4     2     1     0     410
## 9     1     1     1     0    22     4     2     1     0     410
## 10    1     1     1     1    22     4     3     0     0     778
## # ... with 13,534 more rows
```

About the data

- Collected one week before the election
- Nationally representative sample
- Should use post-stratification to control for non-response, but we'll hold off on that for now (See Gelman & Hill, Chapter 14 for more details)

Baseline probability

Let's assume we want to estimate the probability that Bush will be elected.

We could fit a single level model like this

$$\text{bush} \sim \text{Binomial} \left(n = 1, \text{prob}_{\text{bush}=1} = \hat{P} \right)$$
$$\log \left[\frac{\hat{P}}{1 - \hat{P}} \right] = \alpha$$

In code

Can you write the code for the previous model?

```
bush_sl <- glm(bush ~ 1,  
               data = polls,  
               family = binomial(link = "logit"))
```

```
arm::display(bush_sl)
```

```
## glm(formula = bush ~ 1, family = binomial(link = "logit"), data = polls)  
##               coef.est coef.se  
## (Intercept) 0.25      0.02  
## ---  
##    n = 11566, k = 1  
##    residual deviance = 15858.1, null deviance = 15858.1 (difference = 0.0)
```

So, $p_i = \frac{\exp(0.25)}{1 + \exp(0.25)} = 0.56$

01:00

State-level variability

To estimate state-level variability, we just specify a distribution for the intercept variability.

$$\begin{aligned} \text{bush} &\sim \text{Binomial} \left(n = 1, \text{prob}_{\text{bush}=1} = \hat{P} \right) \\ \log \left[\frac{\hat{P}}{1 - \hat{P}} \right] &= \alpha_{j[i]} \\ \alpha_j &\sim N \left(\mu_{\alpha_j}, \sigma_{\alpha_j}^2 \right), \text{ for state } j = 1, \dots, J \end{aligned}$$

Notice we're still specifying that the intercept variability is generated by a *normal* distribution.

What does this variability actually represent?

Variance in the log-odds

Fitting the model

If we're using **{lme4}**, we just swap `lmer()` for `glmer()` and specify the family and link function.

```
library(lme4)
```

```
m0 <- glmer(bush ~ 1 + (1|state),  
            data = polls,  
            family = binomial(link = "logit"))  
arm::display(m0)
```

```
## glmer(formula = bush ~ 1 + (1 | state), data = polls, family = binomial)  
## coef.est  coef.se  
##      0.25      0.06  
##  
## Error terms:  
##   Groups      Name          Std.Dev.  
##   state      (Intercept)  0.34  
##   Residual                        1.00  
## ---  
## number of obs: 11566, groups: state, 49  
## AIC = 15697, DIC = 15450.4  
## deviance = 15571.7
```

Interpretation

- The average log odds of supporting bush was 0.25
- This average varied between states with a standard deviation of 0.34

State-level variation

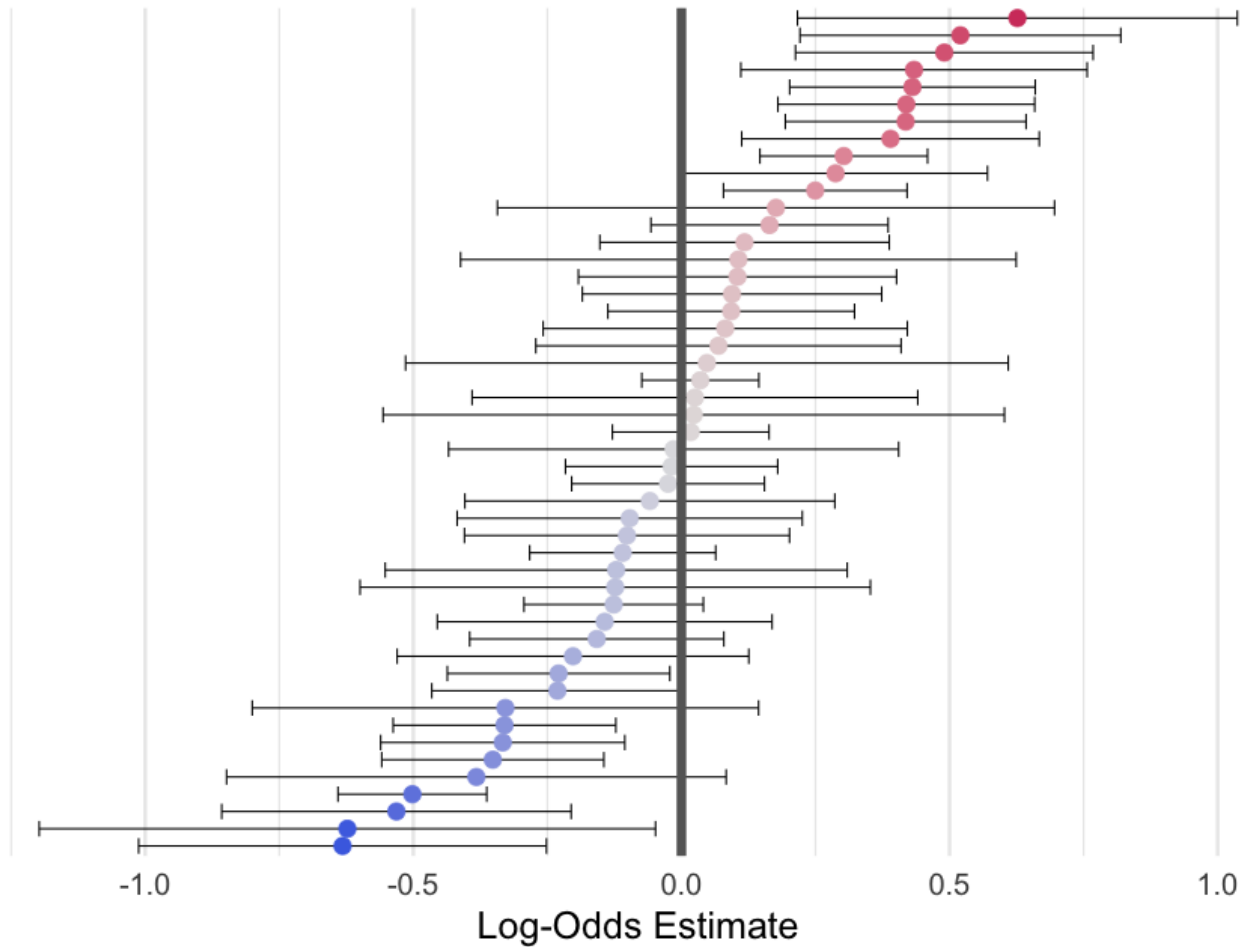
```
library(broom.mixed)
m0_tidied <- tidy(m0, effects = "ran_vals", conf.int = TRUE)
m0_tidied
```

```
## # A tibble: 49 x 8
##   effect    group level term          estimate std.error    conf.low
##   <chr>    <chr> <chr> <chr>          <dbl>      <dbl>      <dbl>
## 1 ran_vals state 1      (Intercept)  0.5201836  0.1526301   0.2210341
## 2 ran_vals state 3      (Intercept)  0.09438107 0.1424292  -0.1847750
## 3 ran_vals state 4      (Intercept)  0.06905650 0.1737493  -0.2714859
## 4 ran_vals state 5      (Intercept)  0.03522421 0.05570977 -0.07396493
## 5 ran_vals state 6      (Intercept)  0.1045418  0.1513626  -0.1921233
## 6 ran_vals state 7      (Intercept) -0.1015410  0.1546380  -0.4046260
## 7 ran_vals state 8      (Intercept) -0.3824312  0.2376744  -0.8482644
## 8 ran_vals state 9      (Intercept) -0.6229779  0.2931982  -1.197636
## 9 ran_vals state 10     (Intercept)  0.3027001  0.07975137  0.1463903
## 10 ran_vals state 11    (Intercept)  0.09282107 0.1173420  -0.1371651
## # ... with 39 more rows, and 1 more variable: conf.high <dbl>
```

Fancified Plot Code

```
m0_tidied %>%
  mutate(level = forcats::fct_reorder(level, estimate)) %>%
  ggplot(aes(estimate, level)) +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_point(aes(color = estimate)) +
  geom_vline(xintercept = 0, color = "gray40", size = 3) +
  labs(x = "Log-Odds Estimate", y = "") +
  colorspace::scale_color_continuous_diverging(palette = "Blue-Red") +
  guides(color = "none") +
  theme(panel.grid.major.y = element_blank(),
        axis.text.y = element_blank())
```

Fancified Plot



Extending the model

Let's add some predictors

- Include **age**, **female** and **black** as fixed effect predictors.
- We should center age so the intercept represents the sample average age.

You try first

03:00

```
polls <- polls %>%
  mutate(age_c = age - mean(age, na.rm = TRUE))

m1 <- glmer(bush ~ age_c + female + black + (1|state),
            data = polls,
            family = binomial(link = "logit"))
arm::display(m1)
```

```
## glmer(formula = bush ~ age_c + female + black + (1 | state),
##       data = polls, family = binomial(link = "logit"))
##               coef.est coef.se
## (Intercept)    0.43      0.07
## age_c         -0.08      0.02
## female        -0.11      0.04
## black         -1.84      0.09
##
## Error terms:
##   Groups      Name              Std.Dev.
##   state      (Intercept)    0.41
##   Residual                                1.00
## ---
## number of obs: 11566, groups: state, 49
## AIC = 15137.1, DIC = 14856.7
## deviance = 14991.9
```

Varying slopes

The average probability of a respondent who was coded **black == 1** who was the average age and non-female supporting Bush was

$$(p_i | \text{age} = 0, \text{female} = 0, \text{black} = 1) = \frac{\exp(0.43 + (-0.08 \times 0) + (-0.11 \times 0) + (-1.84 \times 1))}{1 + \exp(0.43 + (-0.08 \times 0) + (-0.11 \times 0) + (-1.84 \times 1))}$$

$$(p_i | \text{age} = 0, \text{female} = 0, \text{black} = 1) = \frac{\exp(0.43 - 1.84)}{1 + \exp(0.43 - 1.84)}$$

$$(p_i | \text{age} = 0, \text{female} = 0, \text{black} = 1) = \frac{\exp(-1.41)}{1 + \exp(-1.41)}$$

$$(p_i | \text{age} = 0, \text{female} = 0, \text{black} = 1) = \frac{0.24}{1.24}$$

$$(p_i | \text{age} = 0, \text{female} = 0, \text{black} = 1) = 0.19$$

Vary by state?

You try first – try to fit a model that estimates between-state variability in the relation between individuals coded **black**, and their probability of voting for Bush.

```
m2 <- glmer(bush ~ age_c + female + black + (black|state),
            data = polls,
            family = binomial(link = "logit"))
arm::display(m2)
```

```
## glmer(formula = bush ~ age_c + female + black + (black | state),
##       data = polls, family = binomial(link = "logit"))
##               coef.est coef.se
## (Intercept)   0.43      0.07
## age_c         -0.08      0.02
## female        -0.11      0.04
## black         -1.66      0.19
##
## Error terms:
##   Groups      Name      Std.Dev.  Corr
##   state      (Intercept)  0.44
##               black      0.87      -0.46
## Residual                    1.00
## ---
```

01:00

Look at random effects

```
ranef_m2 <- tidy(m2, effects = "ran_vals", conf.int = TRUE) %>%  
  arrange(level)  
ranef_m2
```

```
## # A tibble: 98 x 8  
##   effect    group level term          estimate std.error  conf.low  
##   <chr>    <chr> <chr> <chr>          <dbl>      <dbl>    <dbl>  
## 1 ran_vals state 1    (Intercept)  0.4768807  0.1716175  0.1405165  
## 2 ran_vals state 1    black        1.066314   0.4118905  0.2590240  
## 3 ran_vals state 10   (Intercept)  0.3000900  0.08425460 0.1349540  
## 4 ran_vals state 10   black       -0.1573061  0.3564817 -0.8559974  
## 5 ran_vals state 11   (Intercept)  0.2440973  0.1326607 -0.01591285  
## 6 ran_vals state 11   black       -0.6672948  0.4129706 -1.476702  
## 7 ran_vals state 13   (Intercept) -0.2406066  0.2710096 -0.7717756  
## 8 ran_vals state 13   black        0.2180487  0.8103377 -1.370184  
## 9 ran_vals state 14   (Intercept) -0.04190335 0.09568848 -0.2294493  
## 10 ran_vals state 14   black       -0.3926602  0.3611247 -1.100452  
## # ... with 88 more rows, and 1 more variable: conf.high <dbl>
```


Include fixed effects

```
fe <- data.frame(  
  fixed = fixef(m2),  
  term = names(fixef(m2))  
)  
fe
```

```
##           fixed           term  
## (Intercept) 0.43434106 (Intercept)  
## age_c      -0.08284379 age_c  
## female     -0.10836146 female  
## black      -1.66095113 black
```

```
ranef_m2 <- left_join(ranef_m2, fe)  
ranef_m2 %>%  
  select(level, term, estimate, fixed)
```

```
## # A tibble: 98 x 4  
##   level term           estimate    fixed  
##   <chr> <chr>           <dbl>    <dbl>  
## 1 1      (Intercept) 0.4768807 0.4343411  
## 2 1      black      1.066314 -1.660951  
## 3 10     (Intercept) 0.3000900 0.4343411  
## 4 10     black      -0.1573061 -1.660951
```

Add in fixed effects

```
ranef_m2 <- ranef_m2 %>%  
  mutate(estimate = estimate + fixed)  
ranef_m2
```

```
## # A tibble: 98 x 9  
##   effect    group level term          estimate std.error    conf.low  
##   <chr>    <chr> <chr> <chr>          <dbl>      <dbl>      <dbl>  
## 1 ran_vals state 1    (Intercept)  0.9112217 0.1716175  0.1405165  
## 2 ran_vals state 1    black        -0.5946367 0.4118905  0.2590240  
## 3 ran_vals state 10   (Intercept)  0.7344310 0.08425460 0.1349540  
## 4 ran_vals state 10   black        -1.818257 0.3564817 -0.8559974  
## 5 ran_vals state 11   (Intercept)  0.6784383 0.1326607 -0.01591285  
## 6 ran_vals state 11   black        -2.328246 0.4129706 -1.476702  
## 7 ran_vals state 13   (Intercept)  0.1937344 0.2710096 -0.7717756  
## 8 ran_vals state 13   black        -1.442902 0.8103377 -1.370184  
## 9 ran_vals state 14   (Intercept)  0.3924377 0.09568848 -0.2294493  
## 10 ran_vals state 14   black        -2.053611 0.3611247 -1.100452  
## # ... with 88 more rows, and 2 more variables: conf.high <dbl>, fixed <dbl>
```

Compute log-odds

Compute means for each group – i.e., add the coefficients

```
to_plot <- ranef_m2 %>%  
  group_by(level) %>%  
  mutate(estimate = cumsum(estimate)) %>%  
  ungroup()
```

to_plot

```
## # A tibble: 98 x 9  
##   effect    group level term          estimate std.error    conf.low  
##   <chr>    <chr> <chr> <chr>          <dbl>      <dbl>      <dbl>  
## 1 ran_vals state 1    (Intercept)  0.9112217 0.1716175  0.1405165  
## 2 ran_vals state 1    black        0.3165850 0.4118905  0.2590240  
## 3 ran_vals state 10   (Intercept)  0.7344310 0.08425460 0.1349540  
## 4 ran_vals state 10   black       -1.083826 0.3564817 -0.8559974  
## 5 ran_vals state 11   (Intercept)  0.6784383 0.1326607 -0.01591285  
## 6 ran_vals state 11   black       -1.649808 0.4129706 -1.476702  
## 7 ran_vals state 13   (Intercept)  0.1937344 0.2710096 -0.7717756  
## 8 ran_vals state 13   black       -1.249168 0.8103377 -1.370184  
## 9 ran_vals state 14   (Intercept)  0.3924377 0.09568848 -0.2294493  
## 10 ran_vals state 14   black       -1.661174 0.3611247 -1.100452  
## # ... with 88 more rows, and 2 more variables: conf.high <dbl>, fixed <dbl>
```

Create factor level

```
lev_order <- to_plot %>%  
  filter(term == "(Intercept)") %>%  
  mutate(lev = forcats::fct_reorder(level, estimate))  
  
to_plot <- to_plot %>%  
  mutate(level = factor(level, levels = levels(lev_order$lev)))
```

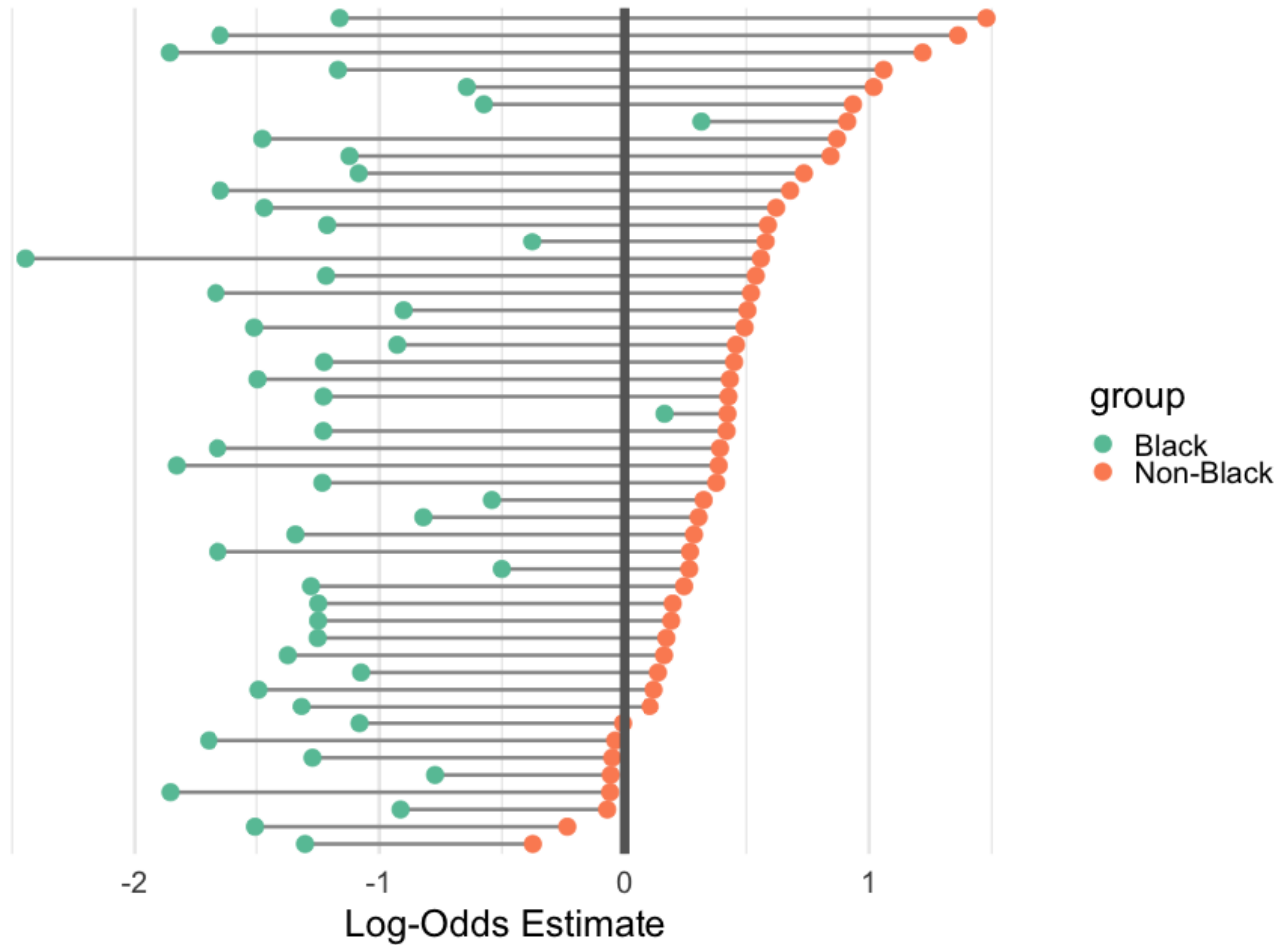
Plot

```
# data transform
to_plot %>%
  mutate(group = ifelse(term == "(Intercept)", "Non-Black", "Black"))

# plot
ggplot(aes(prob, level)) +
  geom_line(aes(group = level), color = "gray60", size = 1.2) +
  geom_point(aes(color = group)) +
  geom_vline(xintercept = 0.5, color = "gray40", size = 3) +

# themeing stuff
labs(x = "Probability Estimate", y = "") +
xlim(0, 1) +
scale_color_brewer(palette = "Set2") +
theme(panel.grid.major.y = element_blank(),
      axis.text.y = element_blank())
```

Plot



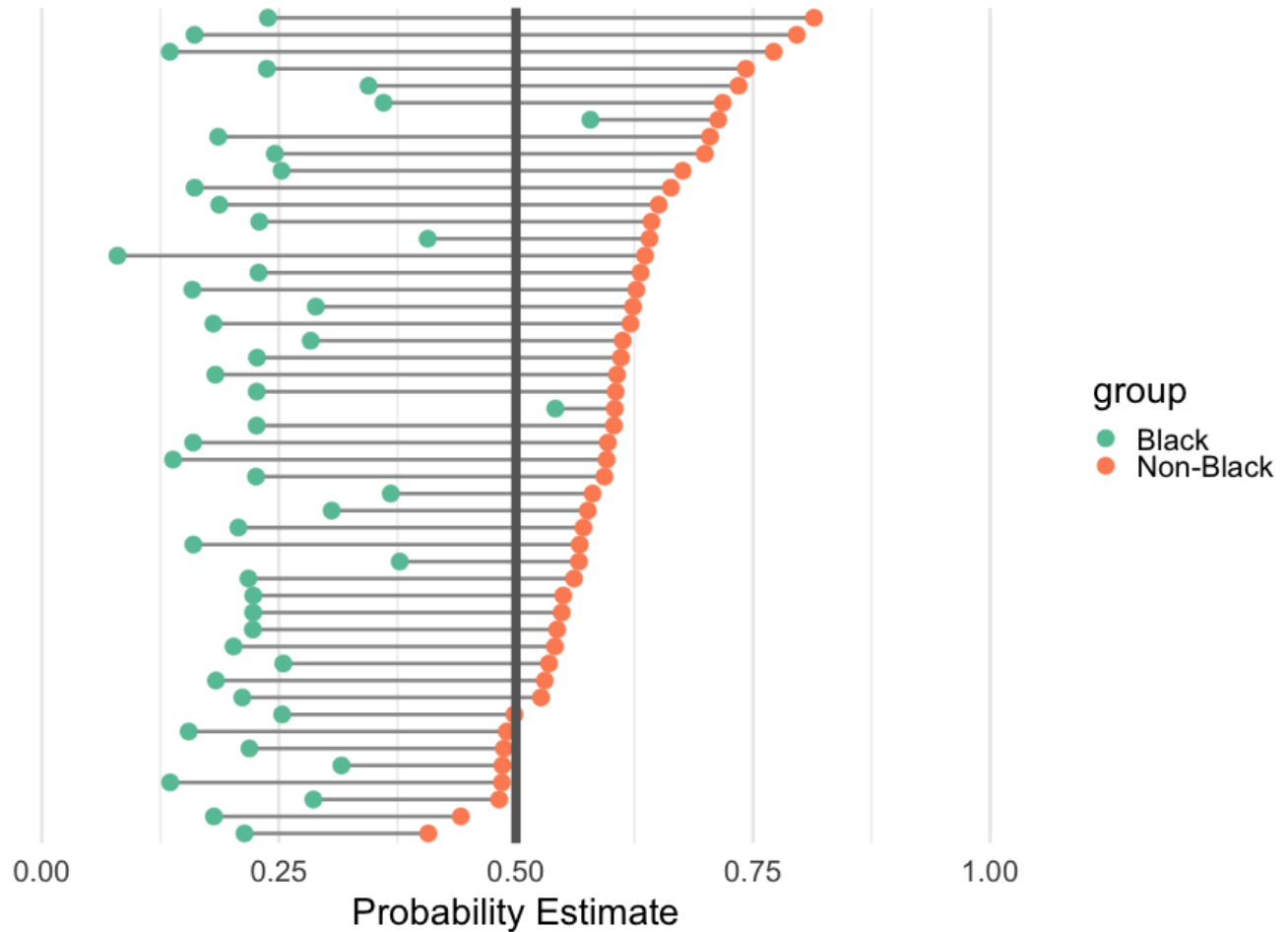
Probability scale

```
# data transform
to_plot %>%
  mutate(
    group = ifelse(term == "(Intercept)", "Non-Black", "Black"),
    prob = exp(estimate)/(1 + exp(estimate))
  ) %>%

# plot
ggplot(aes(prob, level)) +
  geom_line(aes(group = level), color = "gray60", size = 1.2) +
  geom_point(aes(color = group)) +
  geom_vline(xintercept = 0.5, color = "gray40", size = 3) +

# themeing stuff
labs(x = "Probability Estimate", y = "") +
xlim(0, 1) +
scale_color_brewer(palette = "Set2") +
theme(panel.grid.major.y = element_blank(),
      axis.text.y = element_blank())
```

Probability scale



Bayes

Refit

Can you fit the same model we just fit, but with **{brms}**?

You try first

Feel free to just use flat priors

02:00

Flat priors

```
library(brms)
m2_brms <- brm(bush ~ age + female + black + (black|state),
               data = polls,
               family = bernoulli(link = "logit"),
               backend = "cmdstan",
               cores = 4)
```

```
## Warning: Rows containing NAs were excluded from the model.
```

```
##
```

```
-
```

```
\
```

```
|
```

```
/
```

```
-
```

```
\
```

```
|
```

```
/
```

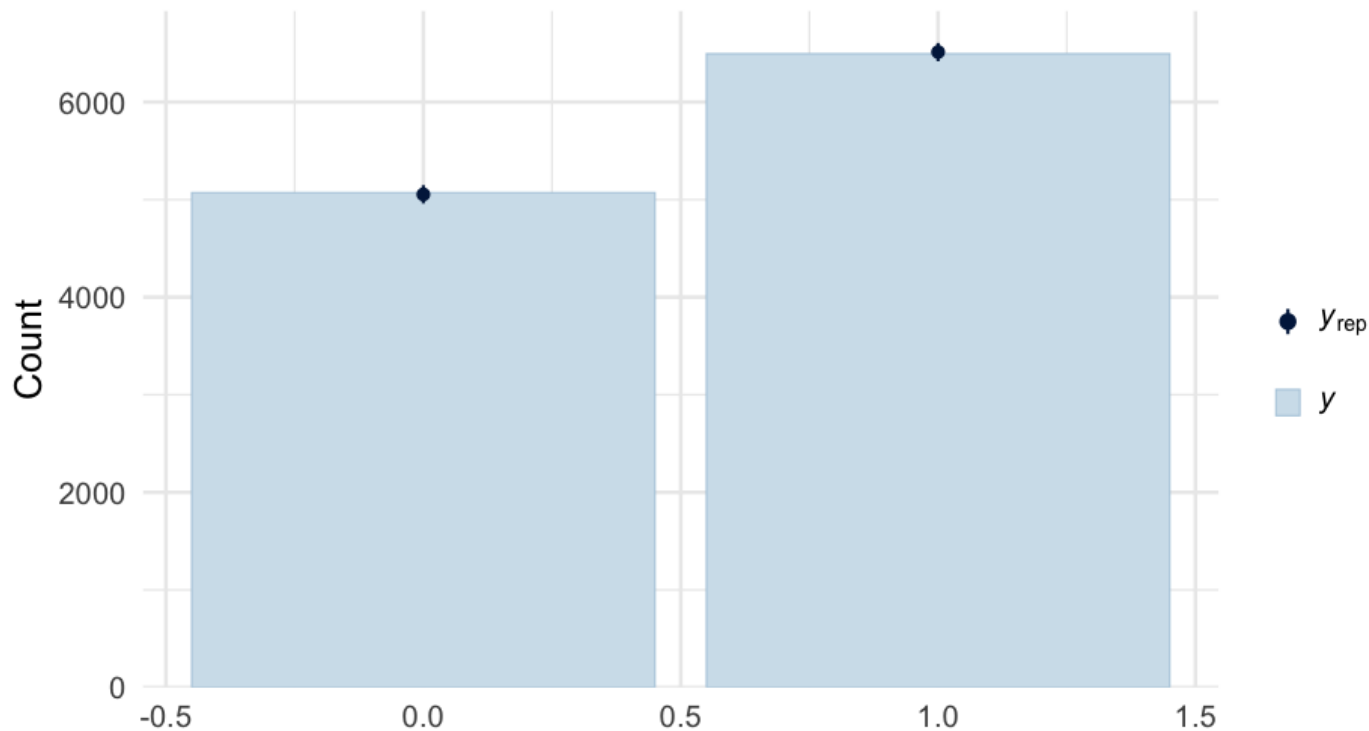
Model summary

```
summary(m2_brms)
```

```
## Family: bernoulli
## Links: mu = logit
## Formula: bush ~ age + female + black + (black | state)
## Data: polls (Number of observations: 11566)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##           total post-warmup samples = 4000
##
## Group-Level Effects:
## ~state (Number of levels: 49)
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)      0.46      0.06    0.36    0.59 1.00      949
## sd(black)           0.96      0.20    0.61    1.41 1.01     1202
## cor(Intercept,black) -0.40      0.19   -0.71    0.02 1.00     1417
##           Tail_ESS
## sd(Intercept)      2013
## sd(black)           2403
## cor(Intercept,black) 2354
##
## Population-Level Effects:
##           Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      0.62      0.09    0.46    0.80 1.00      732     1254
## age            -0.08      0.02   -0.12   -0.05 1.00     5674     3257
## female         -0.11      0.04   -0.19   -0.03 1.00     5243     2793
## black          -1.67      0.22   -2.10   -1.24 1.00     1050    1216887
```

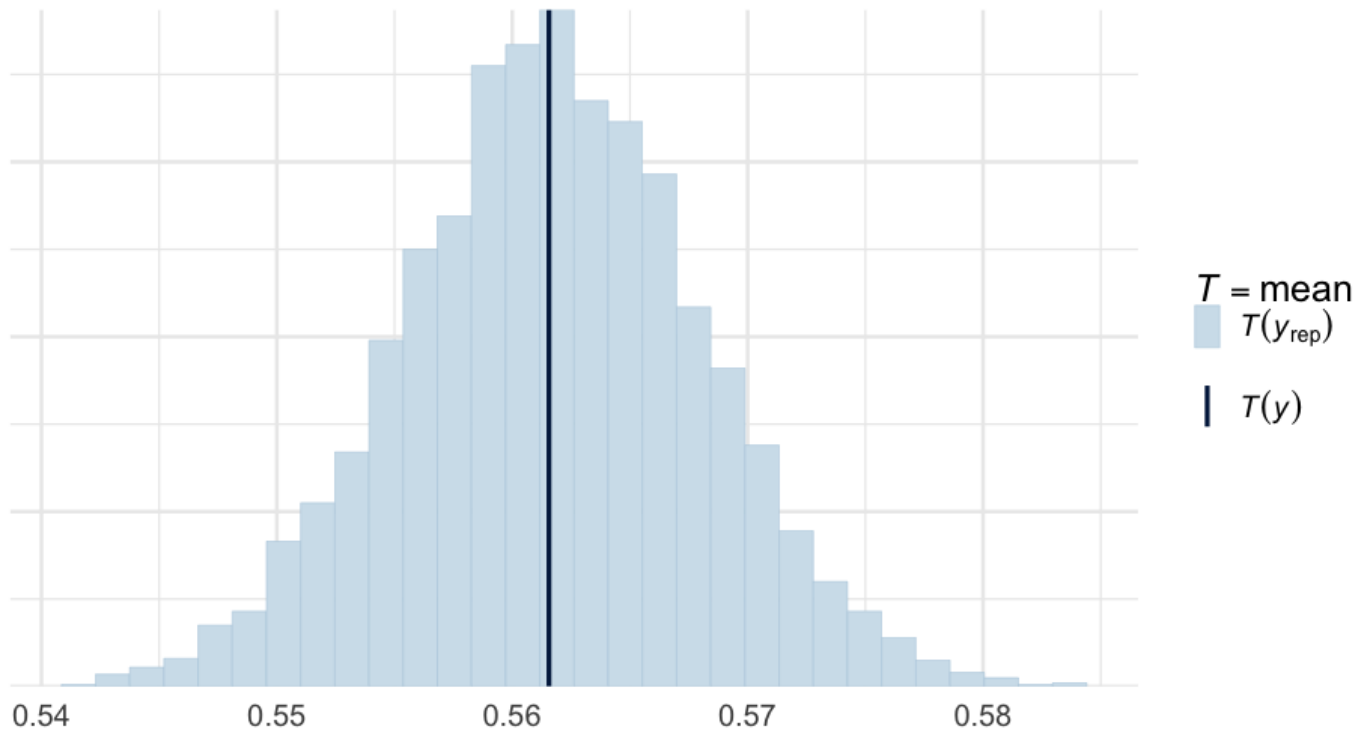
Posterior predictive check

```
pp_check(m2_brms, type = "bars")
```



Posterior predictive check 2

```
pp_check(m2_brms, type = "stat")
```



Next time

- More Bayes for binomial logistic regression
- Plotting Bayes models
- Missing data
- Piece-wise growth models