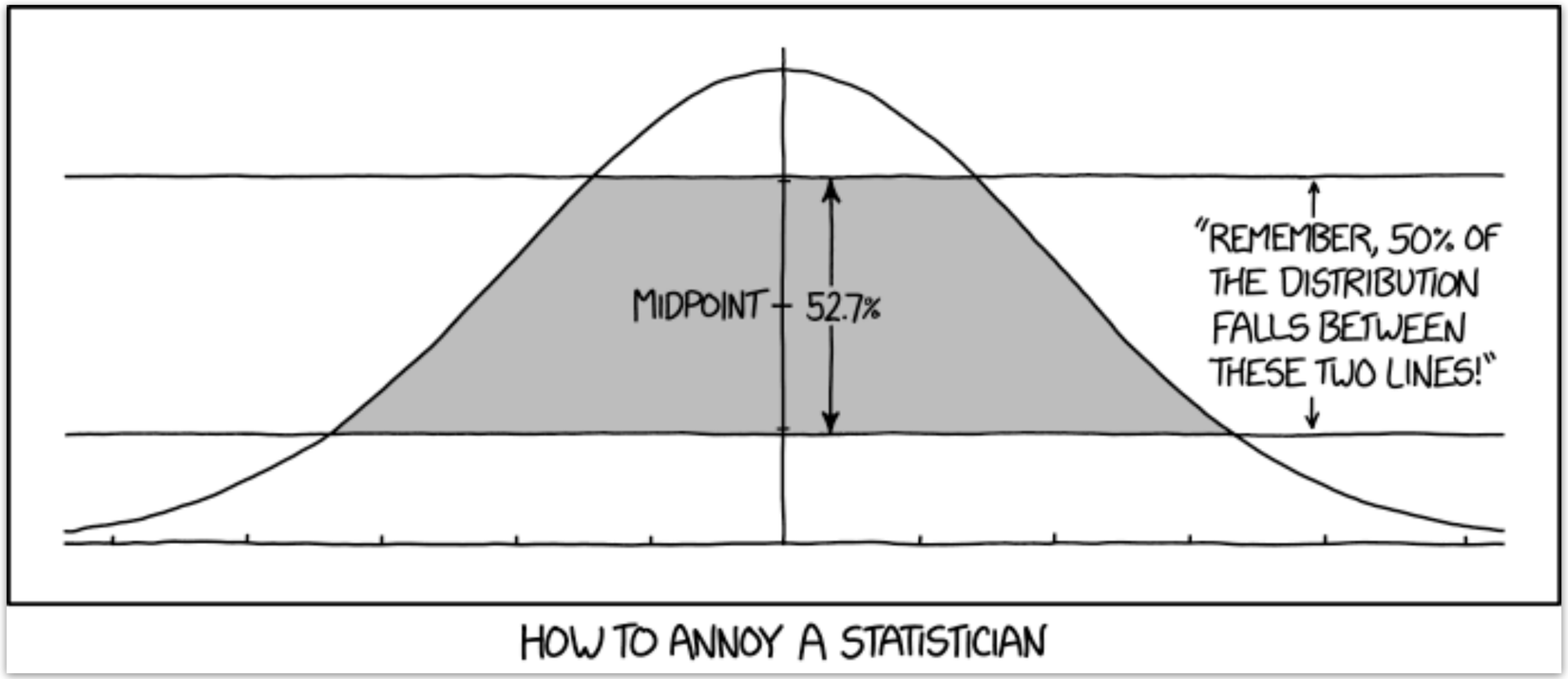
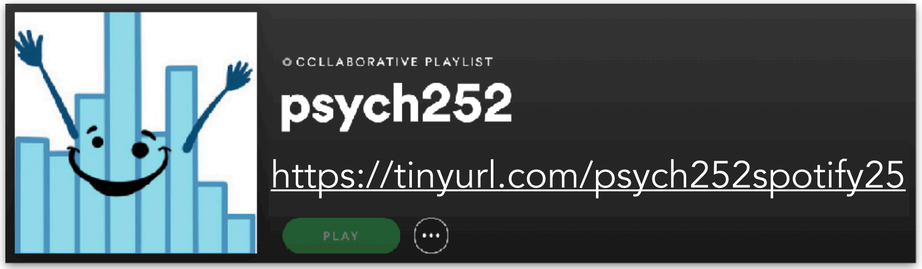
PSYCH 252: Statistical Methods for Behavioral and Social Sciences

# Bayesian data analysis 2



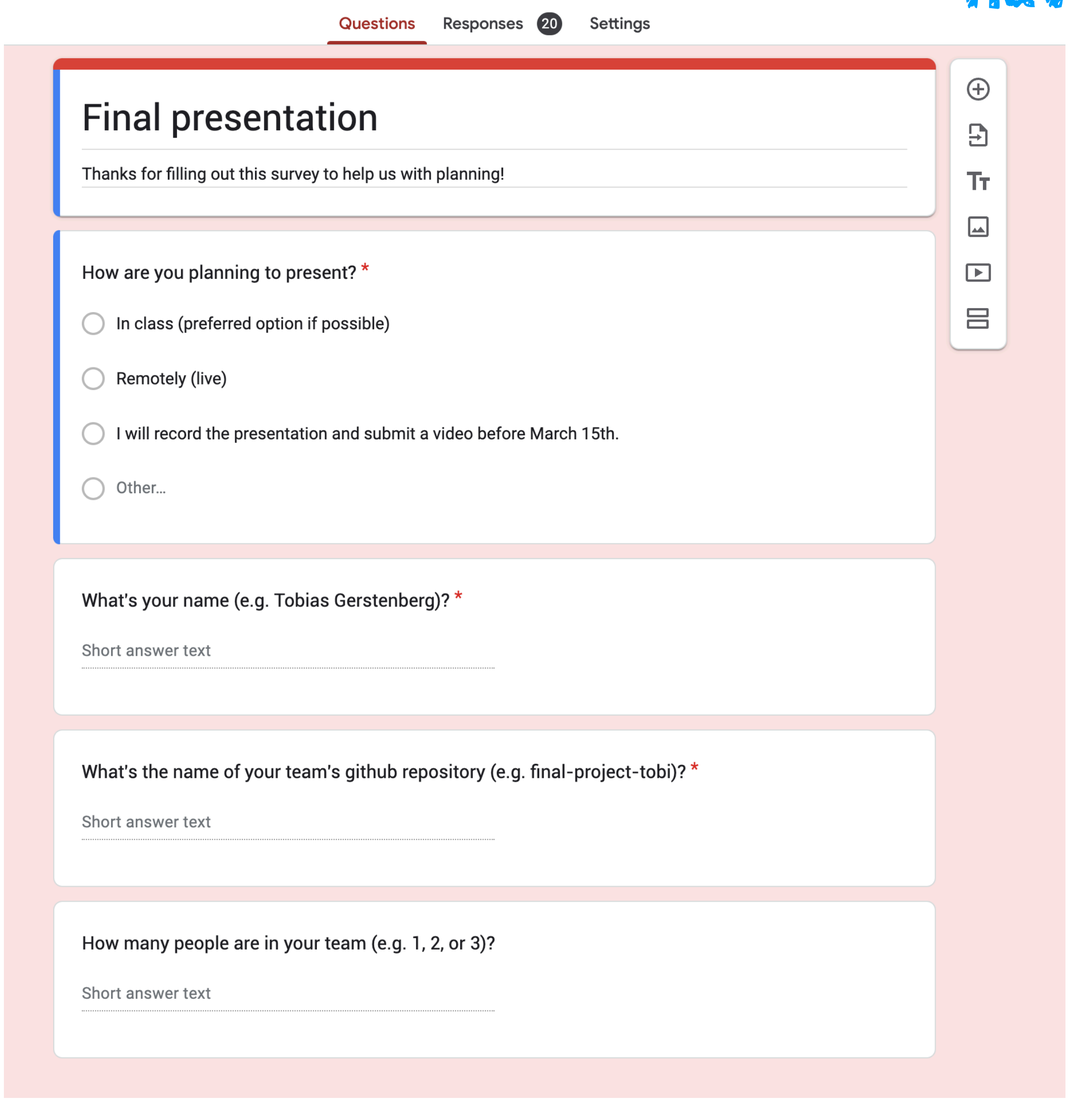




03/05/2025

# Logistics

# Final presentation

please fill out if you haven't already




https://tinyurl.com/psych252presentation25

# Things that came up

# Plan for today

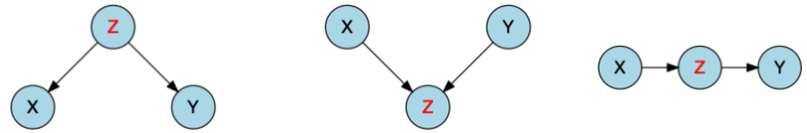
- Quick recap
- Bayesian data analysis
  - Comparison between frequentist and Bayesian data analysis
  - Quick flash from the past
  - Flipping coins
  - What affects the posterior?
  - Ingredients: likelihood, prior, inference
  - Doing Bayesian data analysis

# Quick recap

# Quick recap: Controlling

## Patterns of inference

We want to estimate the (causal) relationship between X and Y




common cause | common effect | causal chain

by controlling for Z we hope to get a better estimate of the relationship between X and Y

## When should I control for variables?

**How can I tell whether two variables are independent?**

Recipe for independence

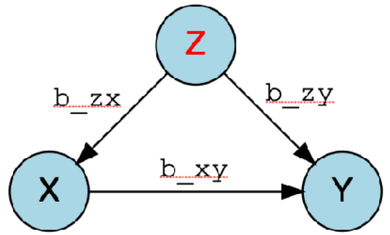
1. Draw the ancestral graph
2. "Moralize" the graph by "marrying" the parents
3. "Disorient" the graph by replacing arrows with edges
4. Delete the givens and their edges
5. Read the answer off the graph

- if variables are **disconnected** they are independent
- if variables are connected (have a path between them) they are not guaranteed to be independent

http://web.mit.edu/jmn/www/6.034/d-separation.pdf 37

## When should I control for variables?

```
1  set.seed(1)
2
3  n = 1000
4  b_zx = 2
5  b_xy = 2
6  b_zy = 2
7  sd = 1
8
9  df = tibble(z = rnorm(n = n, sd = sd),
10             x = b_zx * z + rnorm(n = n, sd = sd),
11             y = b_zy * z + b_xy * x + rnorm(n = n, sd = sd))
```




overestimating X's effect on Y

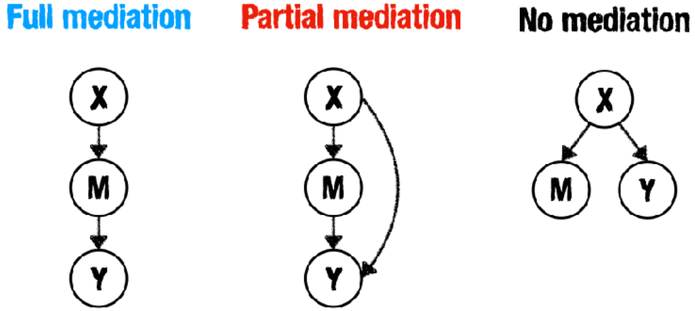
$$Y = b_0 + b_1 \cdot X + e$$

```
1  # without control
2  lm(formula = y ~ x,
3     data = df) %>%
4    summary()
```

```
Call:
lm(formula = y ~ x, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-4.6011 -0.9270 -0.0596  0.9711  4.0454

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.07449    0.04389   0.558    0.577
x            2.82092    0.01890 149.225   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.388 on 998 degrees of freedom
Multiple R-squared:  0.9571,    Adjusted R-squared:  0.9571
F-statistic: 2.227e-04 on 1 and 998 DF,  p-value: < 2.2e-16
```

51

## Underlying causal model



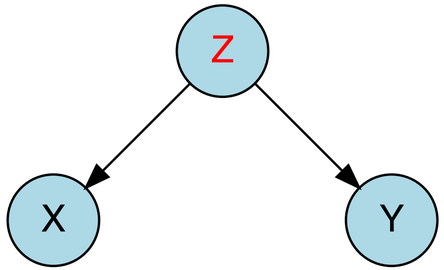

**Full mediation**: When the effect of **X** on **Y** completely disappears, **M** fully mediates between **X** and **Y**.

**Partial mediation**: When the effect of **X** on **Y** still exists, but in a smaller magnitude, **M** partially mediates between **X** and **Y**.

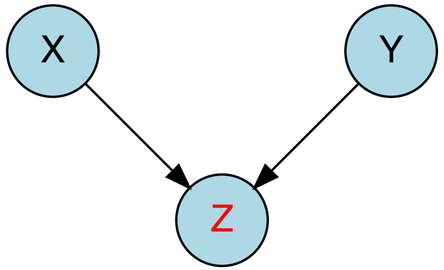
74

# Patterns of inference

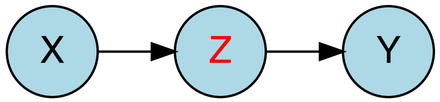
We want to estimate the (causal) relationship between `X` and `Y`




common cause




common effect



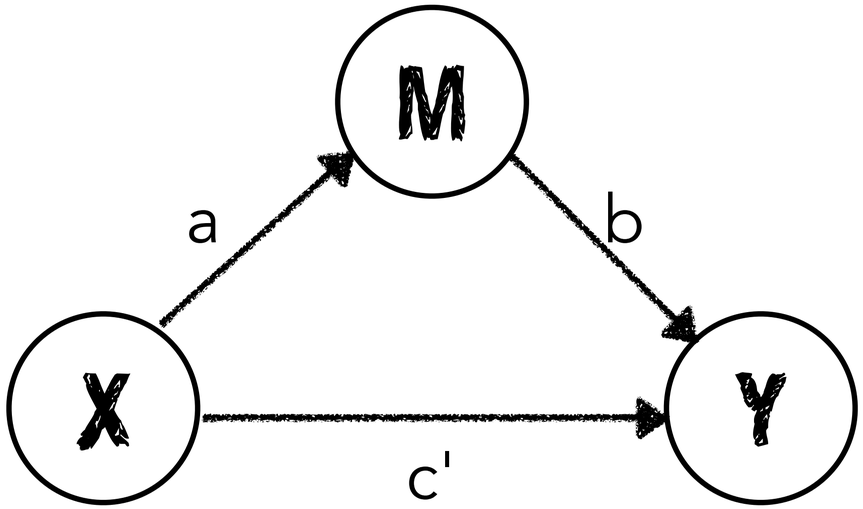

causal chain

by controlling for `Z` we hope to get a better estimate of the relationship between `X` and `Y`

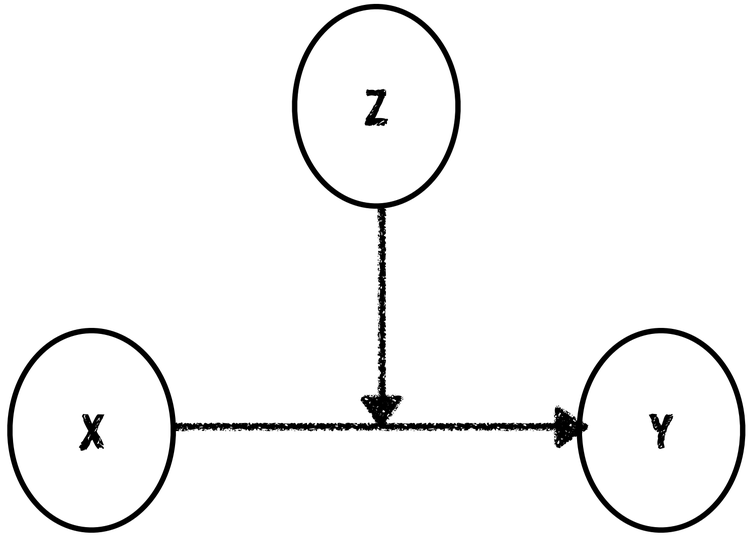
***d***-separation helps us tell apart **good controls** from **bad controls**

*Helpful in the experimental design process - what should be measured?*
*Always requires theory about the underlying causal process*

# Quick recap: Mediation and Moderation




**Mediation** means that the **X** influences the mediator variable **M**, which in turn influences **Y**.




**Moderation** means that the effect of a predictor **X** on outcome **Y** depends on the value of another variable **Z**.

The nature of the relationship between **X** and **Y** depends on **Z**.


NO
MEDIATION

# Quick recap: Bayesian data analysis philosophy

Goal of data analysis: Inference about the world

**Frequentist statistics**

- generate a sampling distribution of the test statistic assuming $H_0$
- compare observed value of the test statistic with the sampling distribution
- reject the $H_0$ if probability of observed value (or more extreme values) is less than α

**Bayesian statistics**

- directly test hypotheses of interest
- define prior over hypotheses **p(H)**
- compute likelihood of the data for each hypothesis **p(D|H)**
- use Bayes' rule to infer the posterior over hypotheses given the data **p(H|D)**

21

**Clue** guide to probability

$$p(B \mid A) = \frac{p(A \mid B) \cdot p(B)}{p(A)}$$

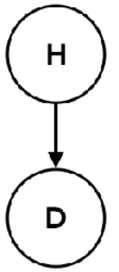
we derived this using the definition of conditional probability

$$\text{posterior}\quad p(H \mid D) = \frac{\overbrace{p(D \mid H)}^{\text{likelihood}} \cdot \overbrace{p(H)}^{\text{prior}}}{p(D)}$$

subjective probability interpretation

$H$ = Hypothesis

$D$ = Data




**formal framework for learning from data**

updating our prior belief $p(H)$, to a posterior belief $p(H \mid D)$ given some data

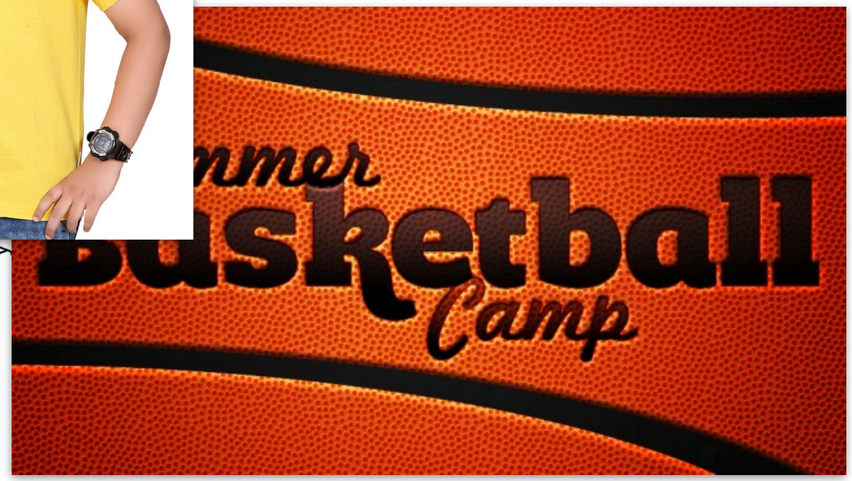
27

# Quick Recap: Summer camp

**prior** $p(\text{chess}) = \frac{1}{3}$

$p(\text{basketball}) = \frac{2}{3}$

**likelihood**


`dnorm(175, mean = 170, sd = 8)` = 0.041

`dnorm(175, mean = 180, sd = 10)` = 0.035

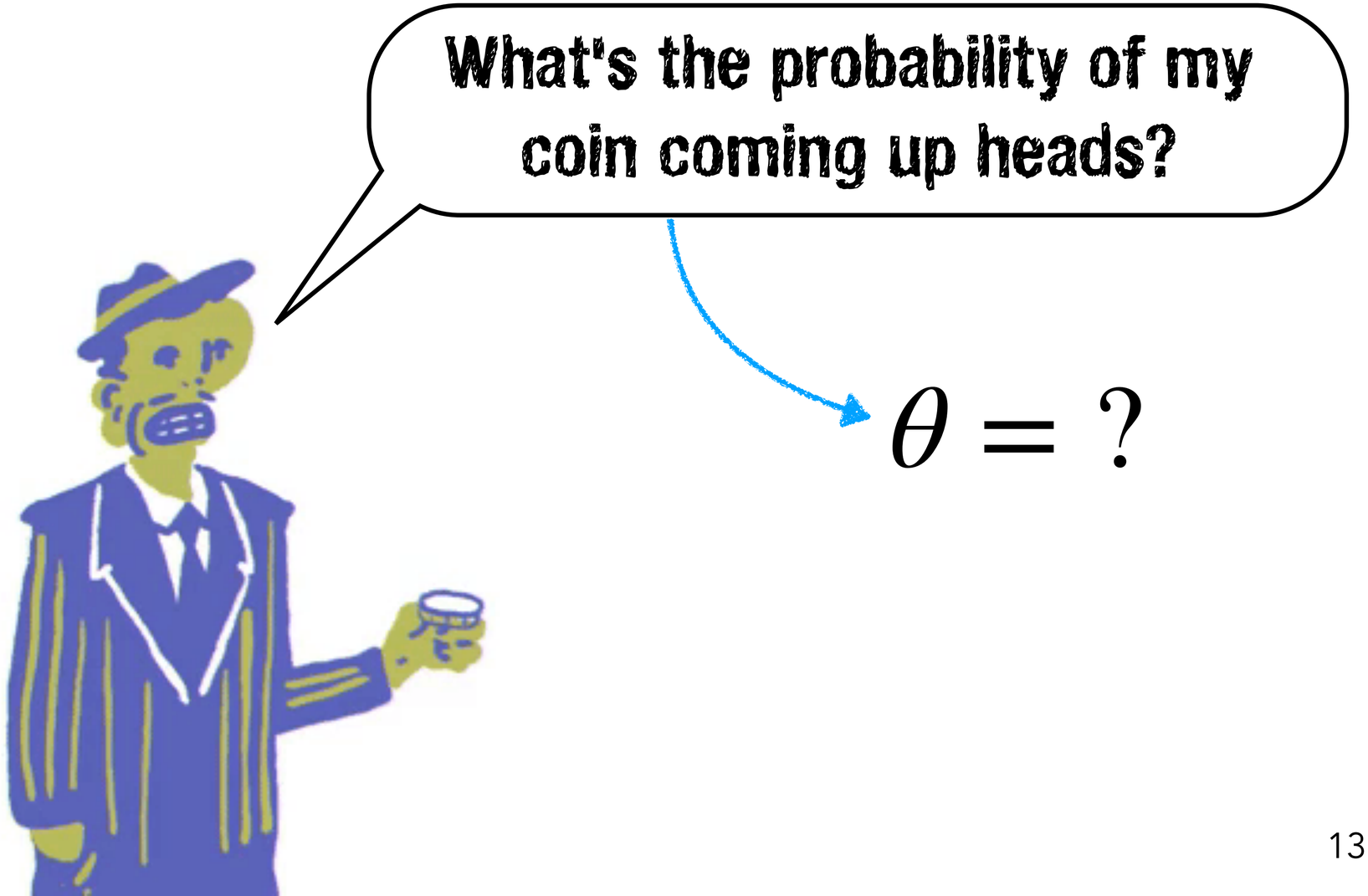
**posterior**

$$p(\text{basketball} \mid 175) = \frac{p(175 \mid \text{basketball}) \cdot p(\text{basketball})}{p(175 \mid \text{basketball}) \cdot p(\text{basketball}) + p(175 \mid \text{chess}) \cdot p(\text{chess})}$$

$$p(\text{basketball} \mid 175) = \frac{0.035 \cdot 2/3}{0.035 \cdot 2/3 + 0.041 \cdot 1/3} \approx 0.63$$

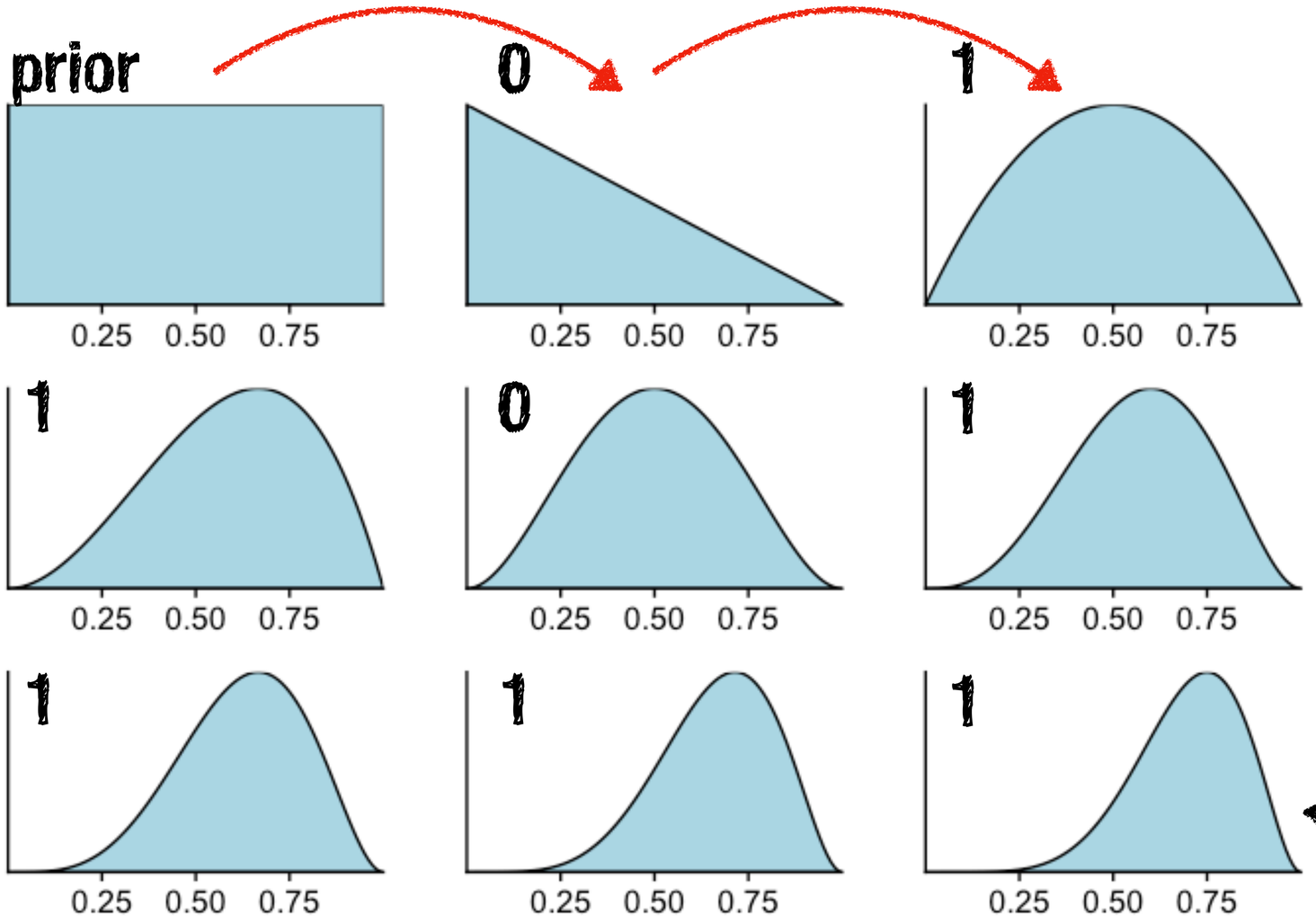
send the kid to the basketball gym!

# Flipping coins



# Learning from data

**How does/should our belief change as evidence comes in?**




**Today's posterior is tomorrow's prior.**

$p(\theta \mid n_{success} = 6, n_{trials} = 8)$

# Coin flip example

**Which coin did I flip?**

## Hypotheses

$\theta = 0.1$


$\theta = 0.5$


$\theta = 0.9$


## Data


**#8 tails, #2 heads**

# Bayesian Recipe

- Hypotheses
- Prior over hypotheses
- Data
- Likelihood of the data given each hypothesis
- Posterior over hypotheses given the data

**+ a healthy dose of Bayes' rule**

$$p(H \mid D) = \frac{p(D \mid H) \cdot p(H)}{p(D)}$$

# Coin flip example

```r
# data
data = rep(0:1, c(8, 2))

# parameters
theta = c(0.1, 0.5, 0.9)

# prior
prior = c(0.25, 0.5, 0.25)

# likelihood
likelihood = dbinom(sum(data == 1), size = length(data), prob = theta)

# posterior
posterior = likelihood * prior / sum(likelihood * prior)
```

8 tails and 2 heads

hypotheses

prior over the hypotheses

binomial distribution

$$p(H \mid D) = \frac{p(D \mid H) \cdot p(H)}{p(D)}$$

law of total probability:

$$p(D) = \sum_{i=1}^{n} p(D \mid H_i) \cdot p(H_i)$$

# Coin flip example

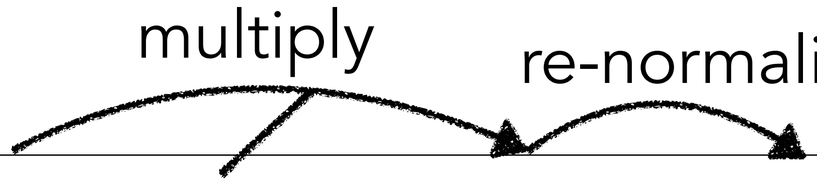
```r
# data
data = rep(0:1, c(8, 2))

# parameters
theta = c(0.1, 0.5, 0.9)

# prior
prior = c(0.25, 0.5, 0.25)

# likelihood
likelihood = dbinom(sum(data == 1), size = length(data), prob = theta)

# posterior
posterior = likelihood * prior / sum(likelihood * prior)
```

8 tails and 2 heads

hypotheses

prior over the hypotheses

binomial distribution




| theta | prior | likelihood | prior_x_like | posterior |
|---|---|---|---|---|
| 0.1 | 0.25 | 0.19 | 0.0475 | 0.69 |
| 0.5 | 0.50 | 0.04 | 0.02 | 0.31 |
| 0.9 | 0.25 | 0.00 | 0.00 | 0.00 |

# Coin flip example

**data: #8 tails, #2 heads**

## Which coin was flipped?

what the model knows before having seen the data


learning by conditioning on the data

what the model knows after having seen the data

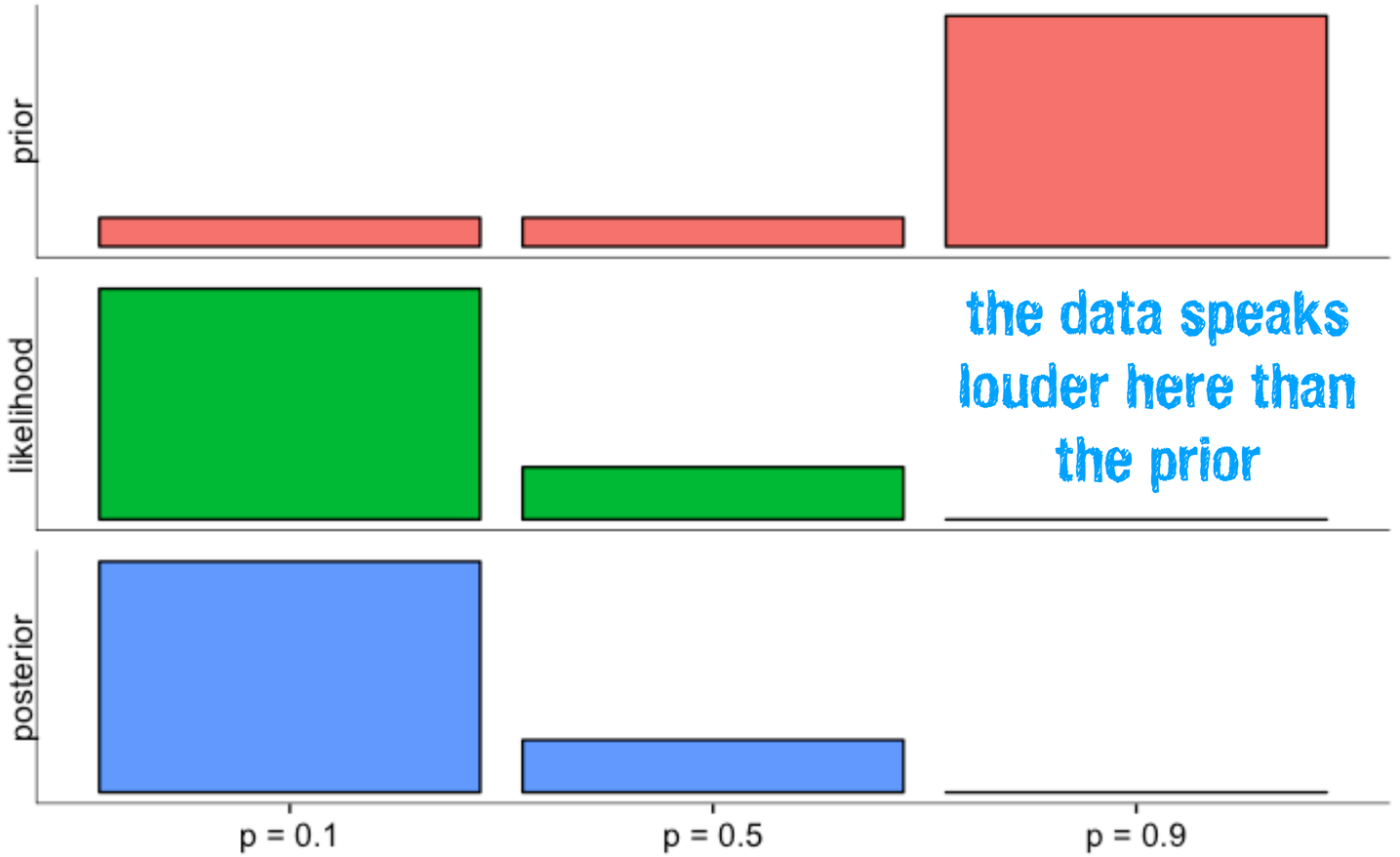
p = 0.1 p = 0.5 p = 0.9

**posterior = multiplicative weighting of prior and likelihood**

# Coin flip example

**data: #8 tails, #2 heads**

**Which coin was flipped?**

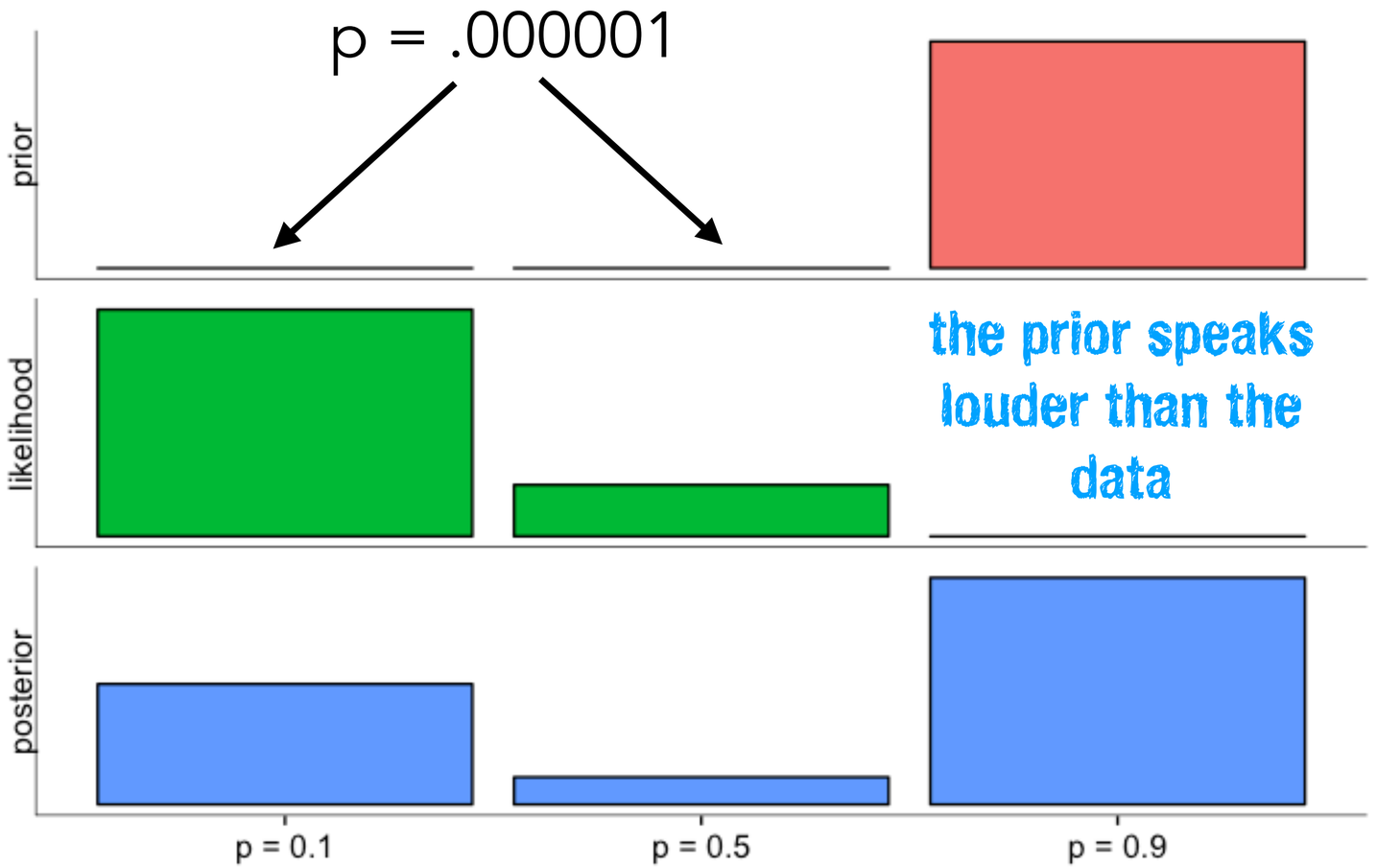



**posterior = multiplicative weighting of prior and likelihood**

# Coin flip example

**data: #8 tails, #2 heads**

## Which coin was flipped?

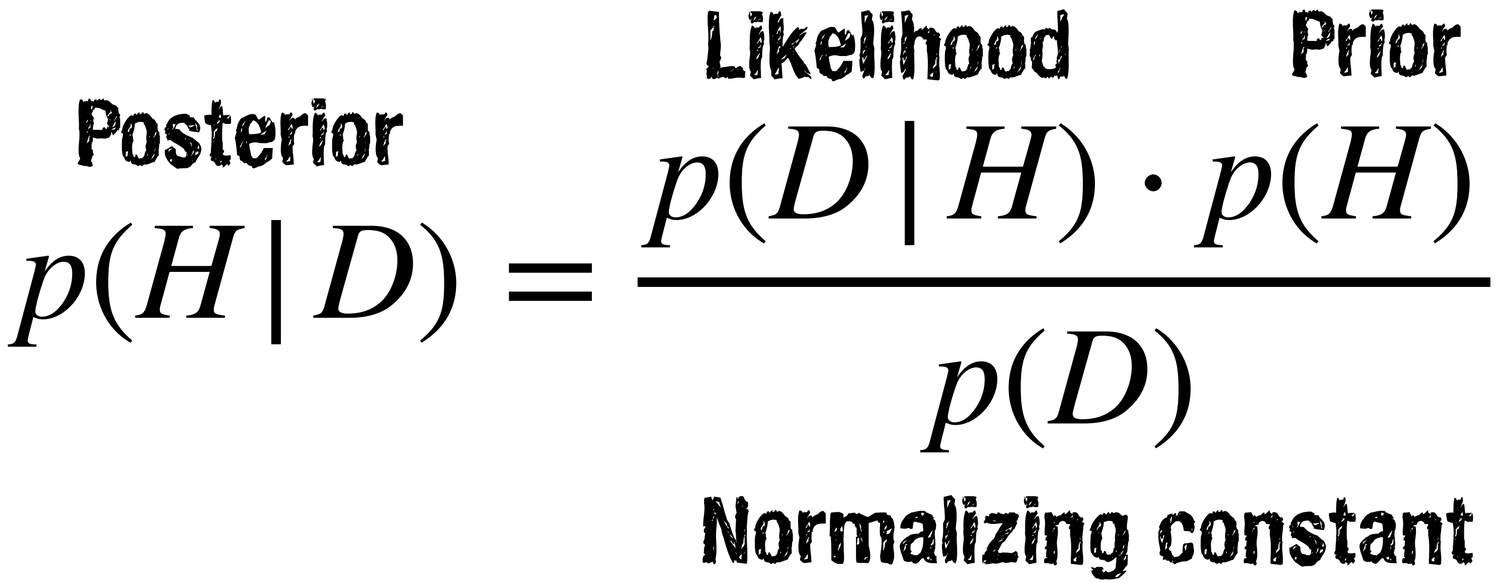



**posterior = multiplicative weighting of prior and likelihood**

# What affects the posterior?

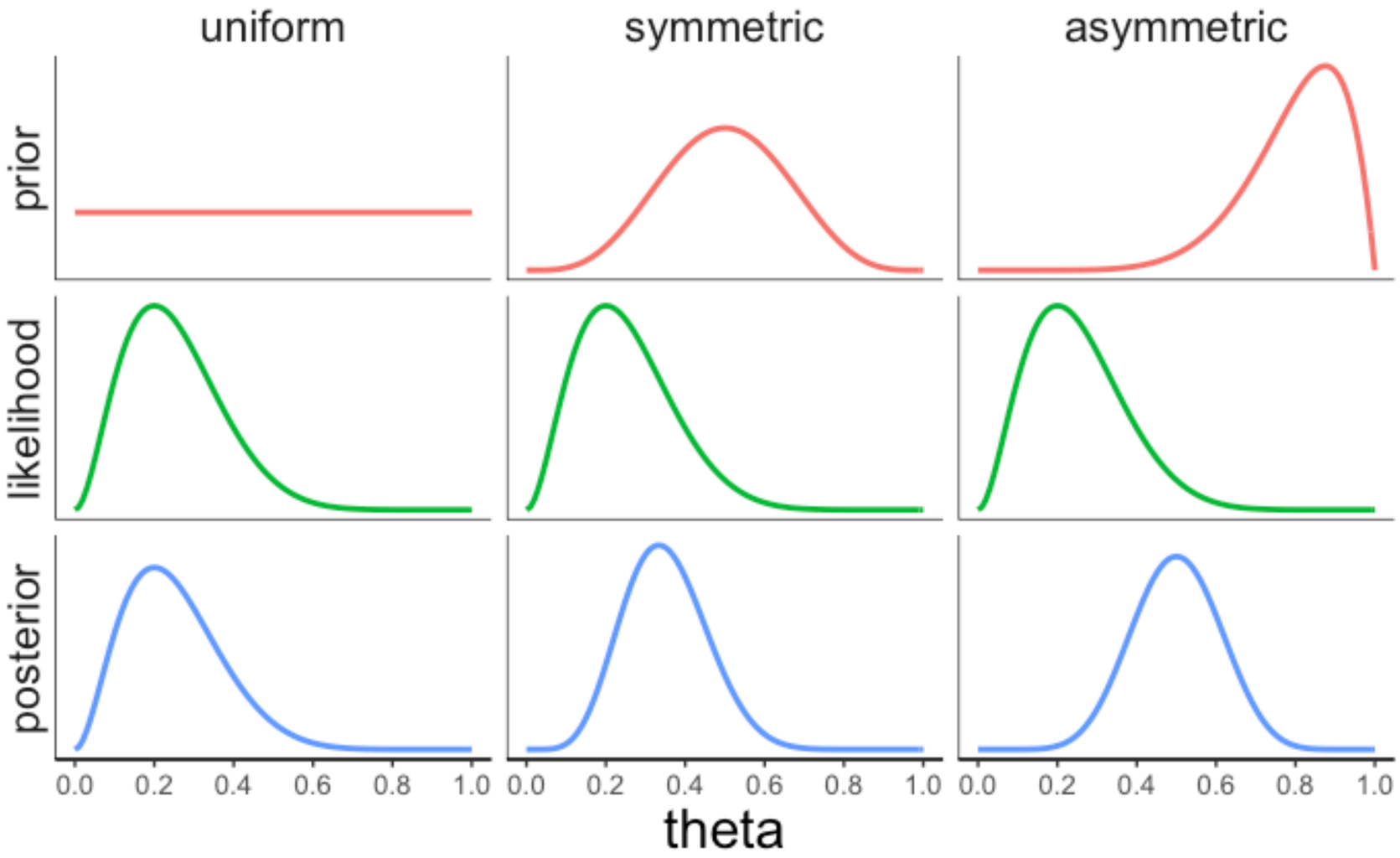
# What affects the posterior?

1. the prior over hypotheses
2. the likelihood of the data given each hypothesis



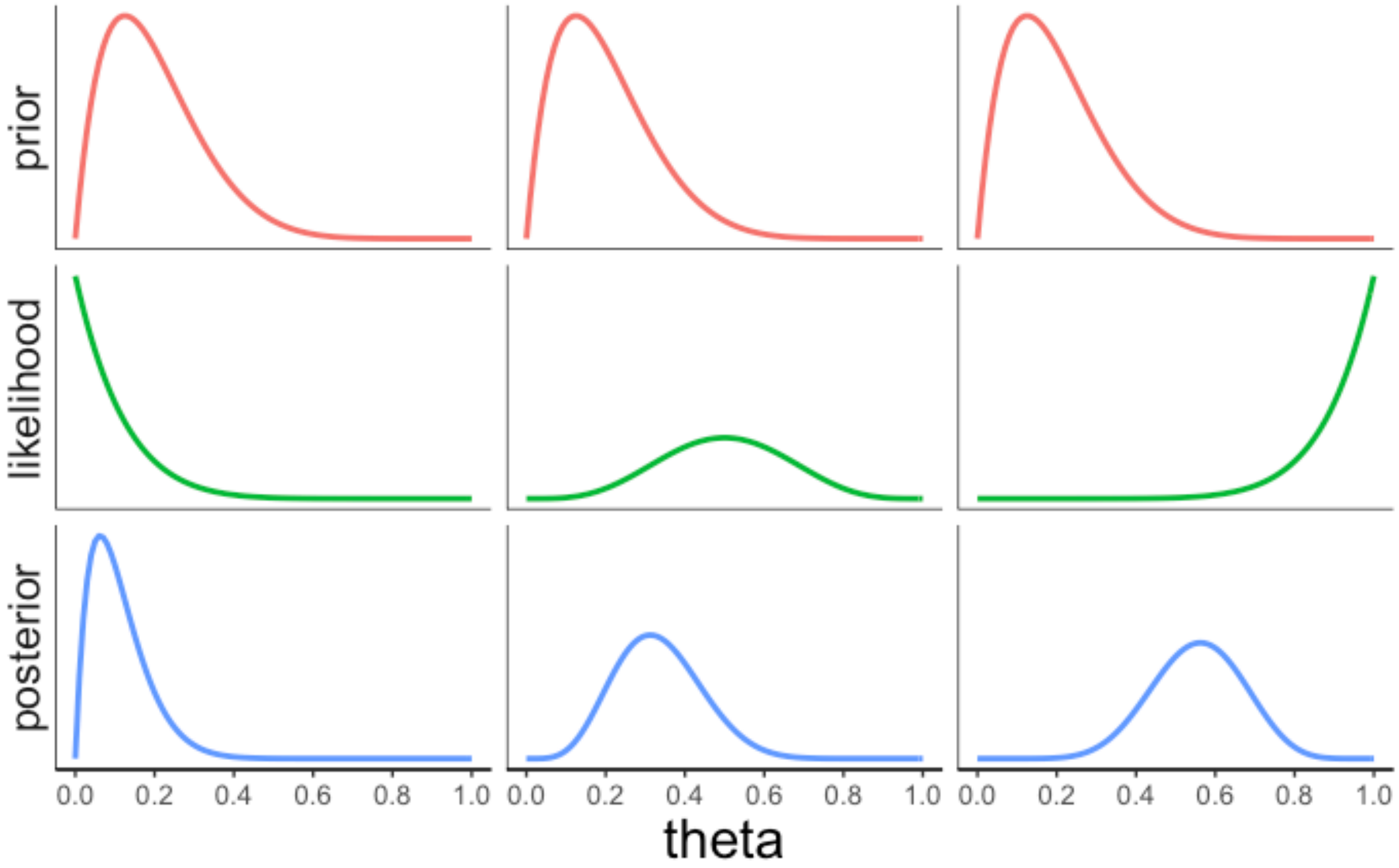
# The effect of the **prior**

same data, different priors

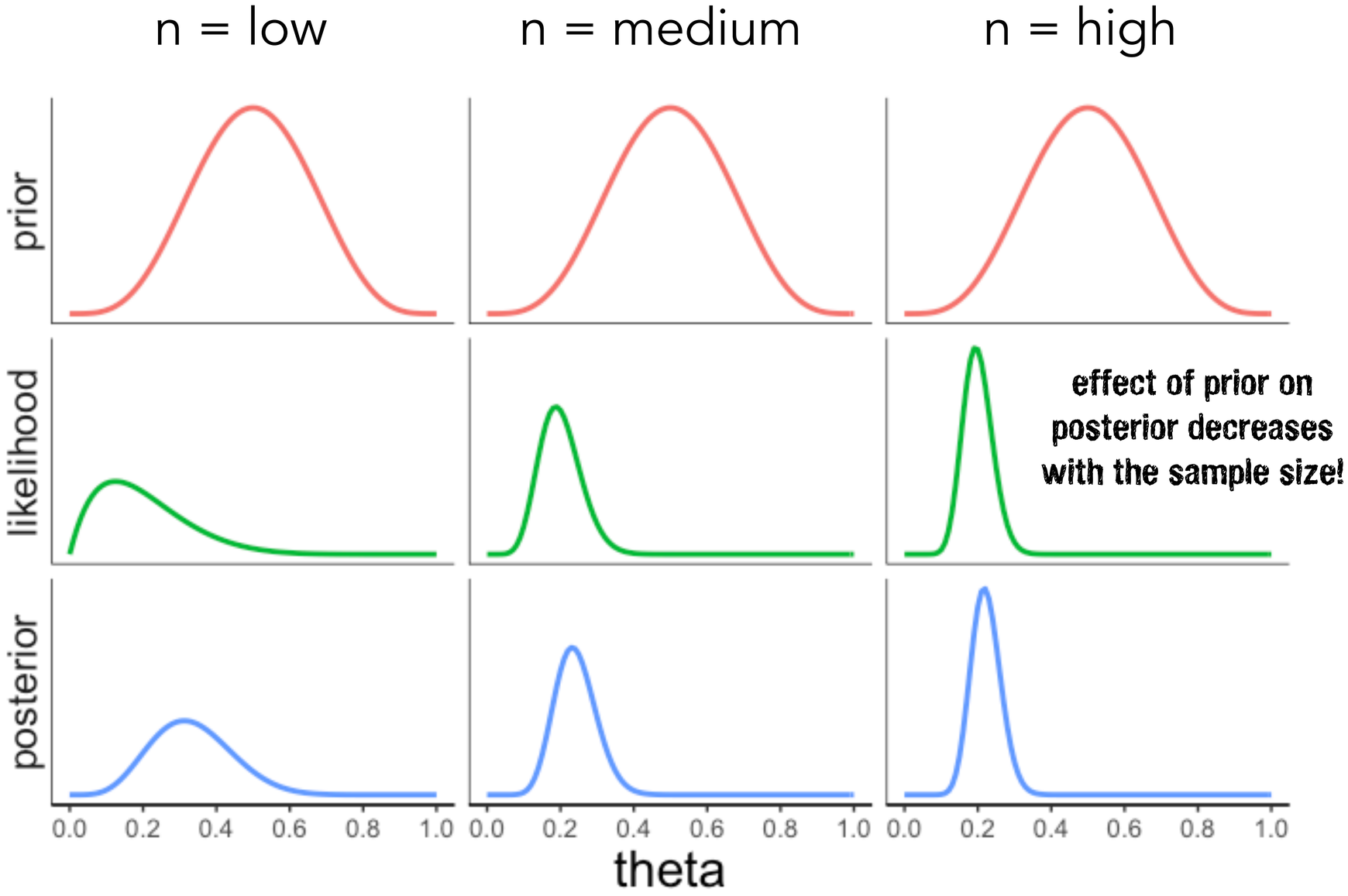


# The effect of the **likelihood**

same prior, different data



# The effect of **sample size**



**Ingredients:** likelihood, prior, inference

# Ingredients

$$\underbrace{p(H \mid D)}_{\text{Posterior}} = \frac{\overbrace{p(D \mid H)}^{\text{Likelihood}} \cdot \overbrace{p(H)}^{\text{Prior}}}{\underbrace{p(D)}_{\text{Normalizing constant}}}$$

Likelihood Prior

Posterior

$$p(H \mid D) = \frac{p(D \mid H) \cdot p(H)}{p(D)}$$

Normalizing constant

# Likelihood

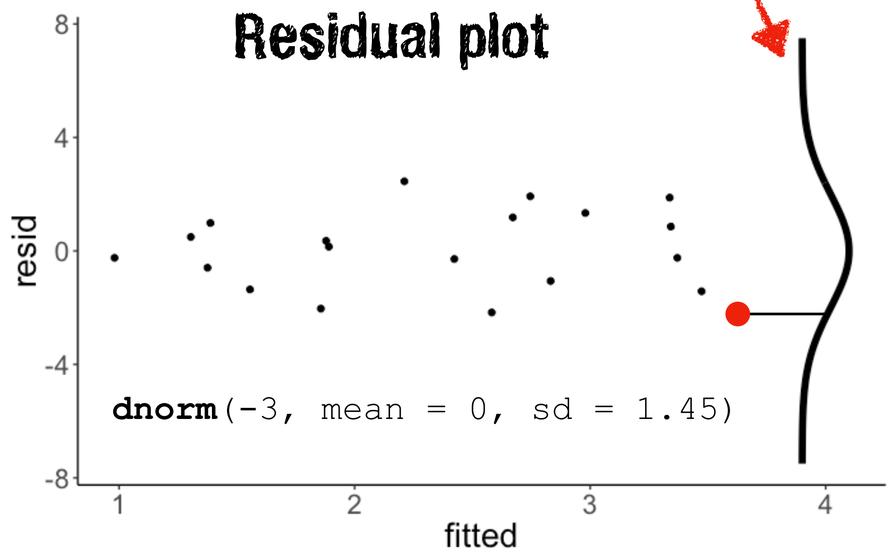
- **What probabilistic model describes best how the data were generated?**
  - What assumptions can you make about the data?
    - What's the nature of your dependent variable (e.g. binary, ordered, continuous)?
  - Does the model re-create the behavior of interest?

# Likelihood

## Gaussian distribution

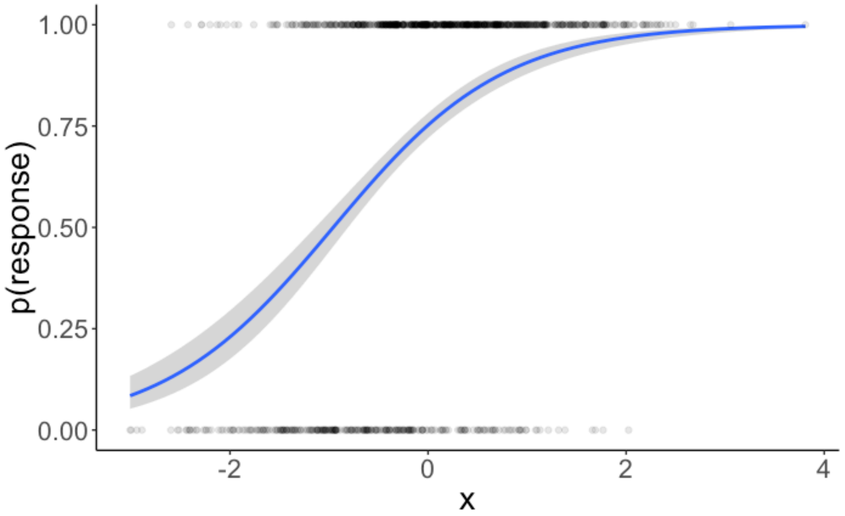
$$Y_i = b_0 + b_1 \cdot x_i + e_i$$

$$e_i \sim \mathcal{N}(0, \sigma)$$

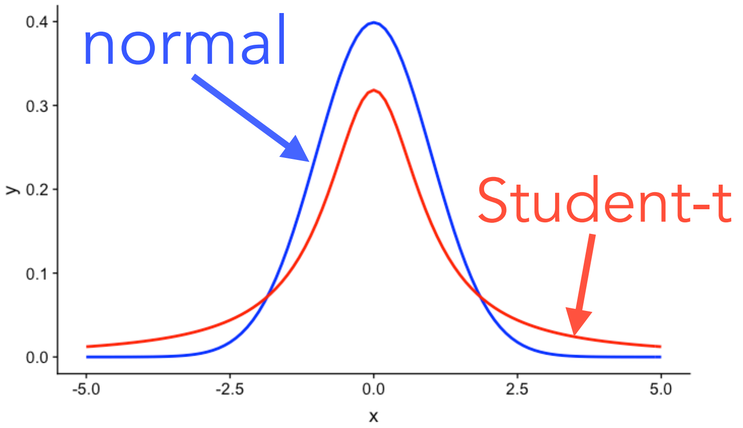



## Binomial distribution

```
1 fit.glm = glm(formula = survived ~ 1 + fare,
2                family = "binomial",
3                data = df.titanic)
```

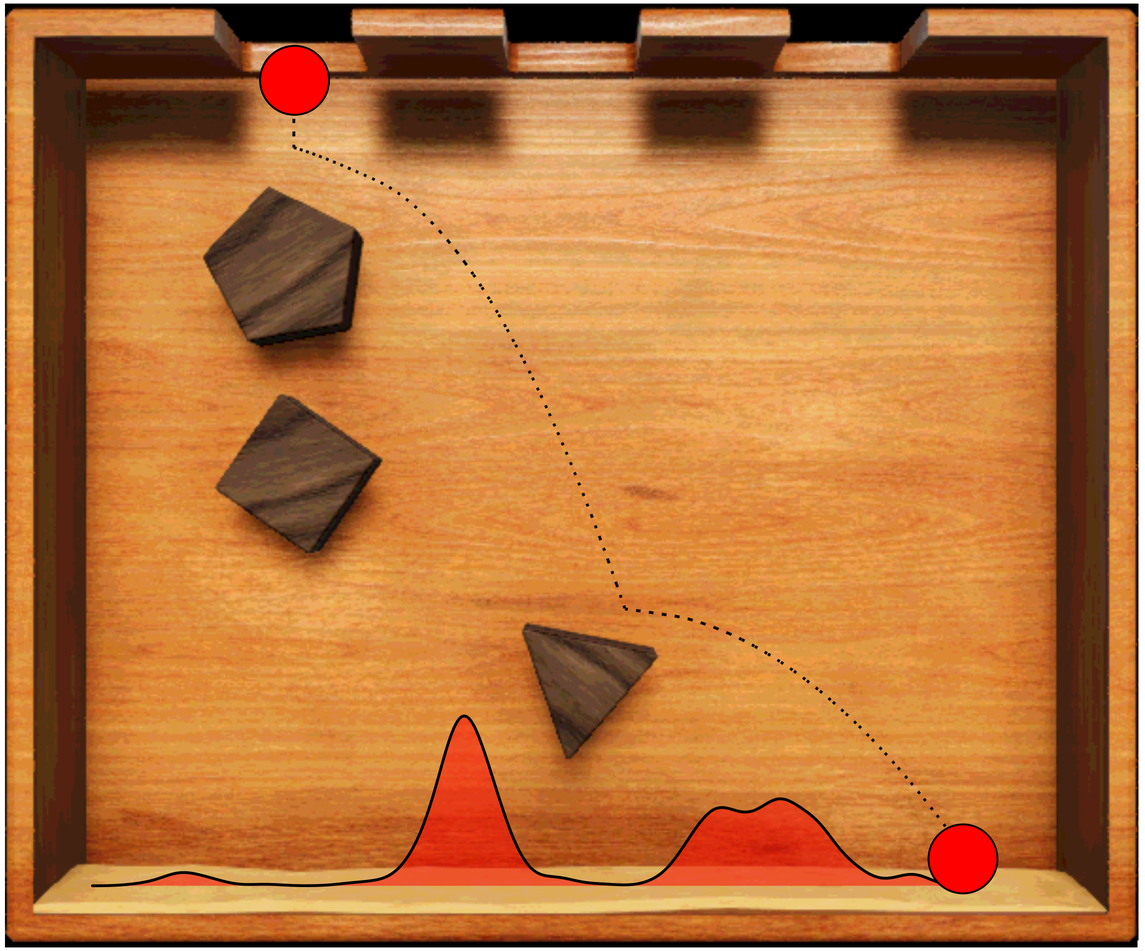


# Likelihood




- **Bernoulli**:
  - binary data
  - a single trial
- **Poisson**: count of discrete events
- **Beta-binomial**: like binomial but probability of success may change across trials
- **Student-*t***:
  - same as Normal
  - handles greater variability in the data (distribution has **fat tails**)
- **...**

# Prediction: Where will the ball land?

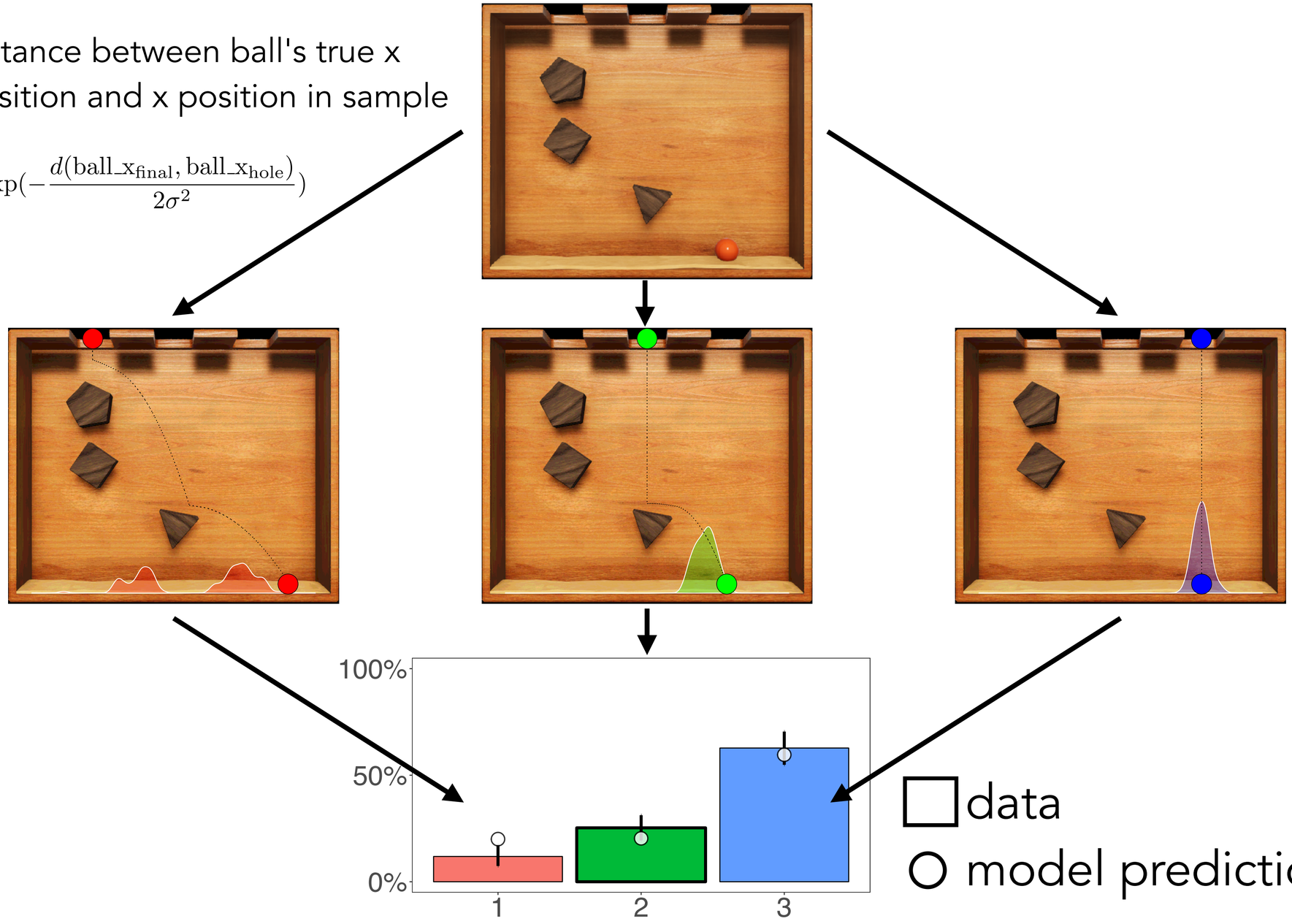


Aggregated responses

# Inference: In which hole was the ball dropped?

distance between ball's true x position and x position in sample

$$\exp\left(-\frac{d(\text{ball\_x}_{\text{final}}, \text{ball\_x}_{\text{hole}})}{2\sigma^2}\right)$$



# Prior

Likelihood Prior

Posterior

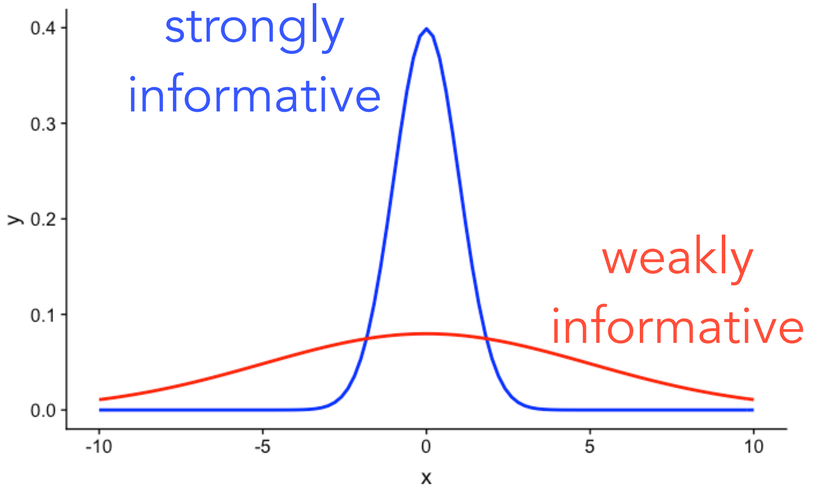
$$p(H \mid D) = \frac{p(D \mid H) \cdot p(H)}{p(D)}$$

Normalizing constant

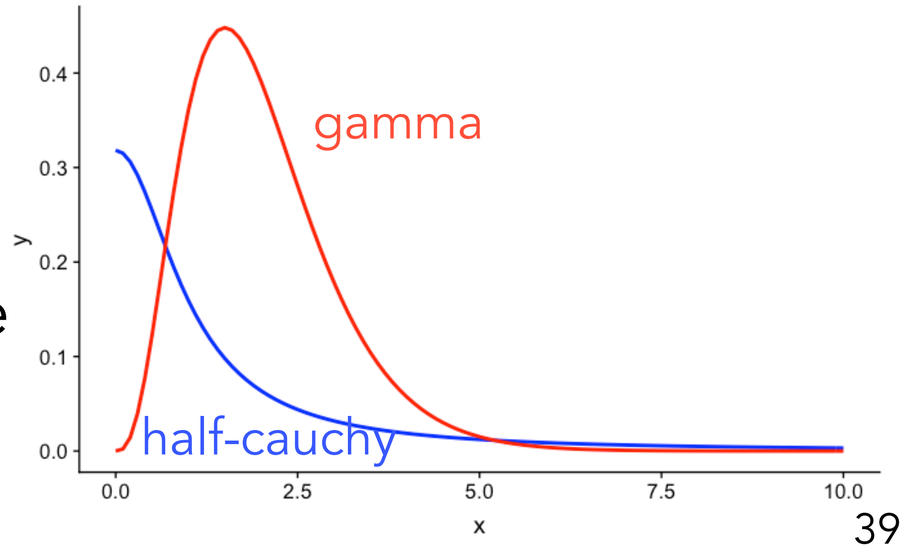
# Prior

- **uniform**:
  - continuous or discrete
  - bounded between minimun and maximum
- **gaussian:**
  - sd determines how informative the prior is
- **gamma, half-cauchy**:
  - for parameters we know are positive

for **beta coefficients** in a regression




for **standard deviation** of the Gaussian



# Inference

Likelihood Prior

Posterior

$$p(H \mid D) = \frac{p(D \mid H) \cdot p(H)}{p(D)}$$

**Normalizing constant**

the devil is in the denominator …

# Doing Bayesian inference

## Discrete hypothesis space

$$p(H \mid D) = \frac{p(D \mid H) \cdot p(H)}{\sum_{i=1}^{n} p(D \mid H_i) \cdot p(H_i)}$$

sum over all possibilities

## Continuous hypothesis space

$$p(H \mid D) = \frac{p(D \mid H) \cdot p(H)}{\int_{-\infty}^{\infty} p(D \mid H_i) \cdot p(H_i) dH_i}$$

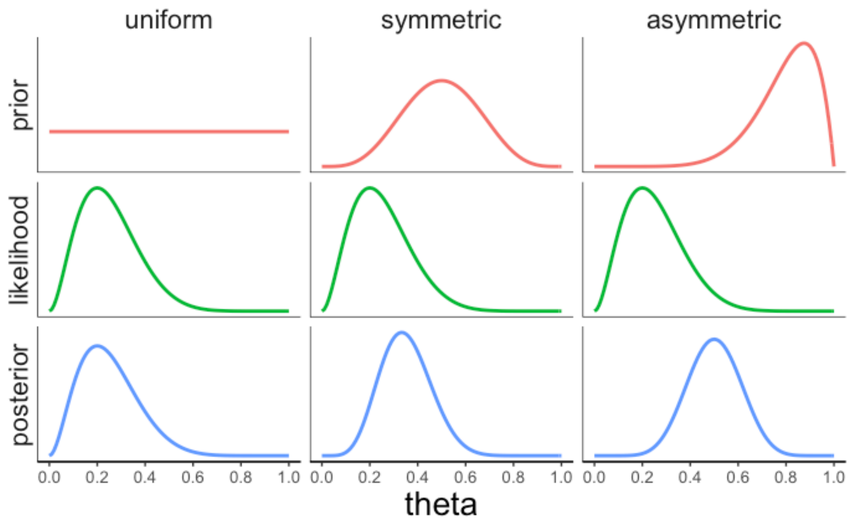
integral over all possibilities

# Discretizing the parameters

```r
# grid
theta = seq(0, 1, 0.01)

# data
data = rep(0:1, c(8, 2))

# calculate posterior
df.prior = tibble(theta = theta,
                  prior_uniform = dbeta(theta, shape1 = 1, shape2 = 1),
                  prior_normal = dbeta(theta, shape1 = 5, shape2 = 5),
                  prior_biased = dbeta(theta, shape1 = 8, shape2 = 2)) %>%
  pivot_longer(cols = -theta,
               names_to = "prior_index",
               values_to = "prior") %>%
  mutate(likelihood = dbinom(sum(data == 1),
                             size = length(data),
                             prob = theta)) %>%
  group_by(prior_index) %>%
  mutate(posterior = likelihood * prior / sum(likelihood * prior)) %>%
  ungroup() %>%
  pivot_longer(cols = -c(theta, prior_index),
               names_to = "index",
               values_to = "value")
```

100 discrete values

**for 3 variables, we would already need 1 Mio combinations**




## The curse of dimensionality

# Inference via sampling

- we cannot directly calculate the probability of the posterior (because it might have a pretty weird shape)
- **but**: we can draw random samples from the posterior
- we can then use our data wrangling and visualization skills to make inferences based on these samples

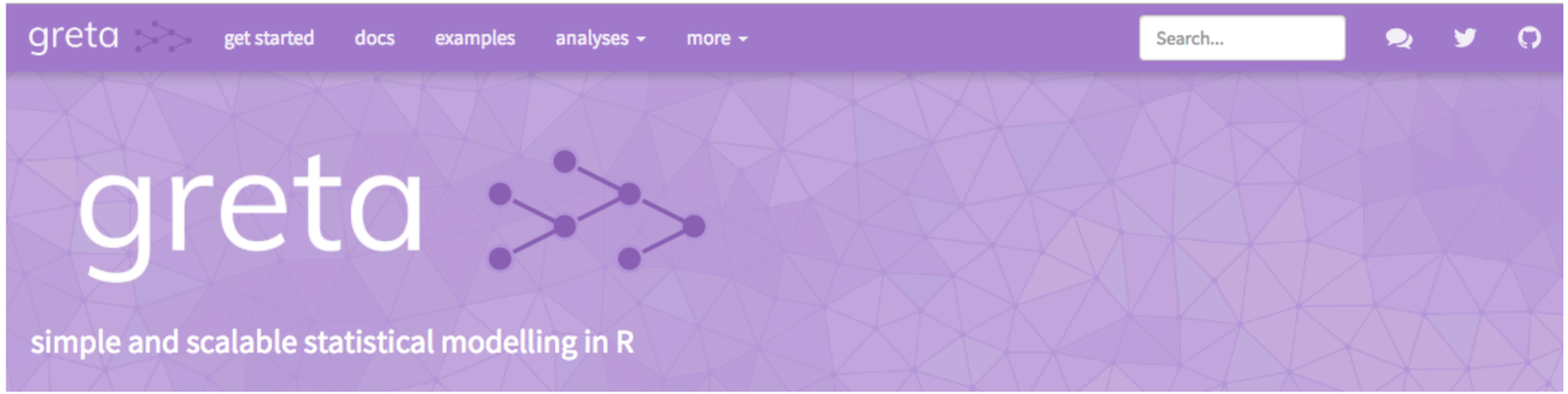
**It's as if ...**

we don't have **`pnorm()`** but we do have **`rnorm()`**

# Inference via sampling

- Bayesian data analysis is becoming more popular because:
  - computers are getting more powerful
  - inference techniques are getting better
  - software packages become easier to use

# Doing Bayesian data analysis with greta

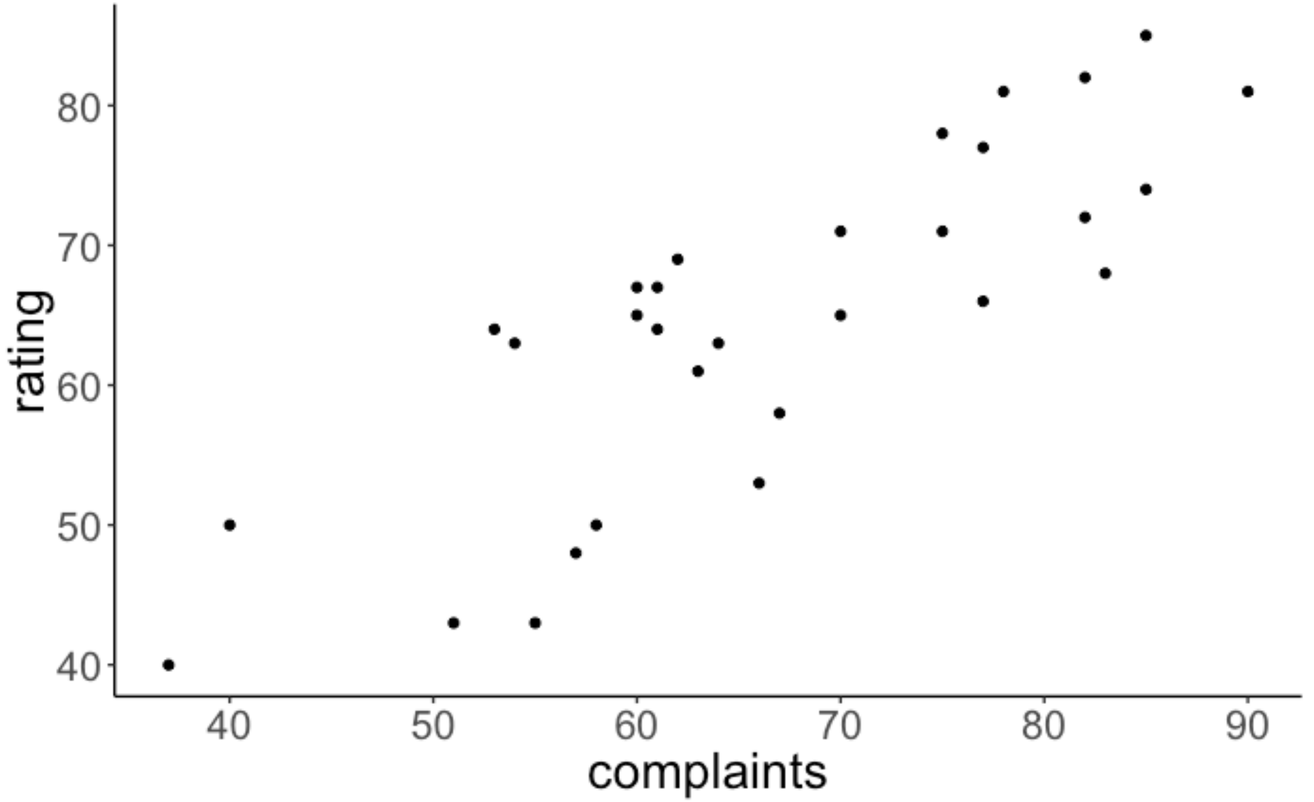



- let's us write Bayesian models directly in R with a simple syntax
- uses Tensorflow to implement Hamiltonian Monte Carlo sampling (a fast inference algorithm ...)

**unfortunately doesn't work on Apple silicone (e.g. M1)**

# Attitude data set

**What's the relationship between how well an employee handles complaints and their overall rating?**



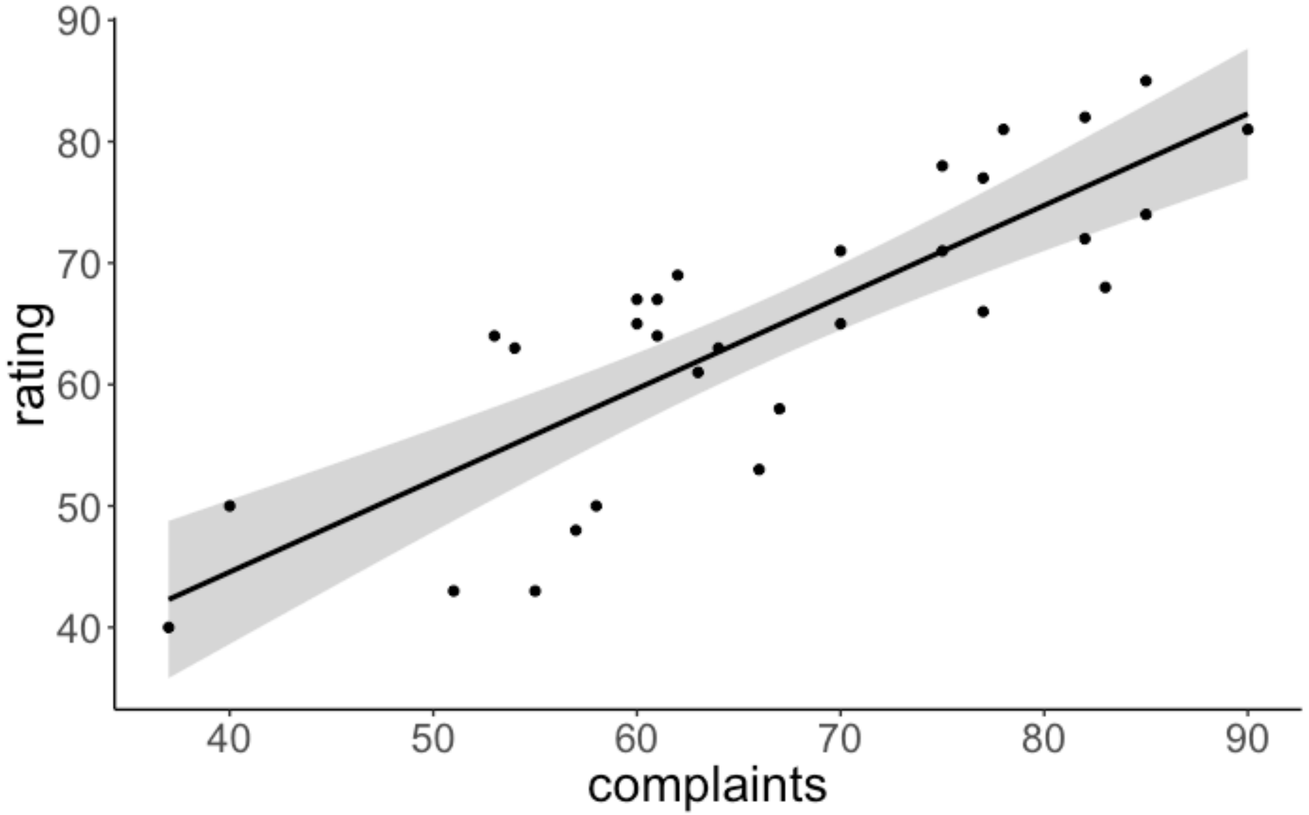
# Frequentist analysis

# Frequentist analysis

```r
# fit model
fit = lm(formula = rating ~ 1 + complaints,
         data = df.attitude)

# print summary
fit %>% summary()
```

```
Call:
lm(formula = rating ~ 1 + complaints, data = df.attitude)

Residuals:
     Min       1Q   Median       3Q      Max
-12.8799  -5.9905   0.1783   6.2978   9.6294

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.37632    6.61999   2.172   0.0385 *
complaints   0.75461    0.09753   7.737 1.99e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.993 on 28 degrees of freedom
Multiple R-squared:  0.6813,    Adjusted R-squared:  0.6699
F-statistic: 59.86 on 1 and 28 DF,  p-value: 1.988e-08
```

# Visualize model predictions




Best-fitting regression line with confidence interval

# Bayesian analysis

# Model specification

```
 1 library("greta")
 2 library("tidybayes")
 3
 4 # variables & priors
 5 b0 = normal(0, 10)
 6 b1 = normal(0, 10)
 7 sd = cauchy(0, 3, truncation = c(0, Inf))
 8
 9 # linear predictor
10 mu = b0 + b1 * attitude$complaints
11
12 # observation model (likelihood)
13 distribution(attitude$rating) = normal(mu, sd)
14
15 # define the model
16 m = model(b0, b1, sd)
```

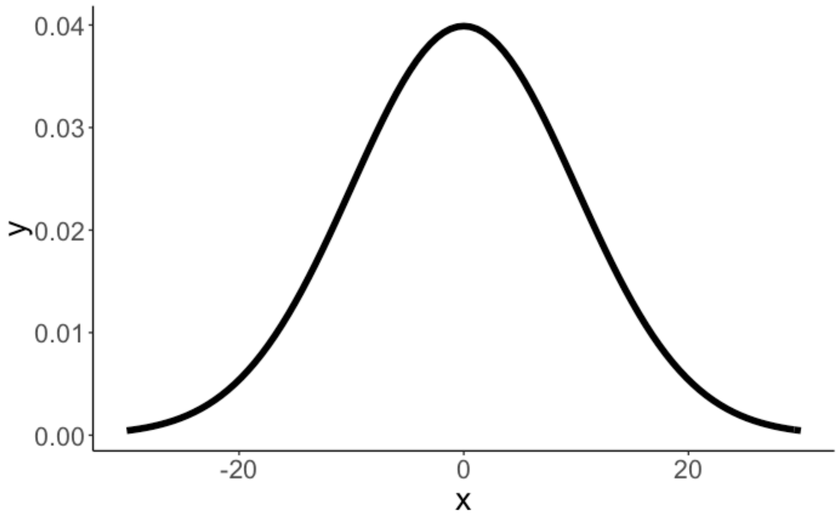
priors

linear combination

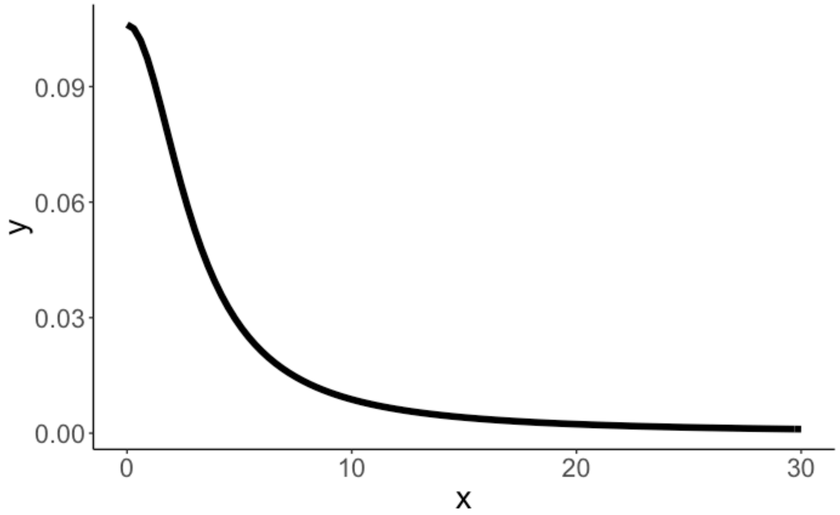
Gaussian likelihood

build the model

# Priors




**Gaussian prior on intercept and coefficient**

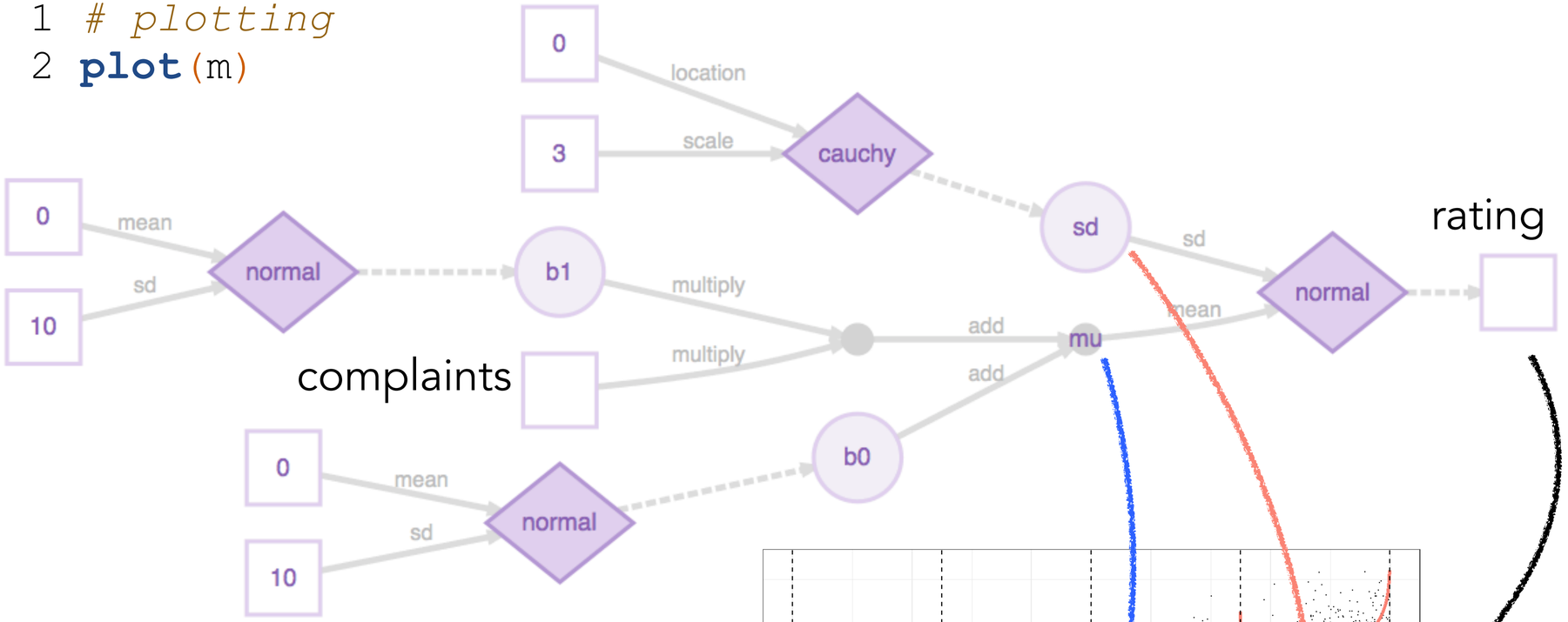


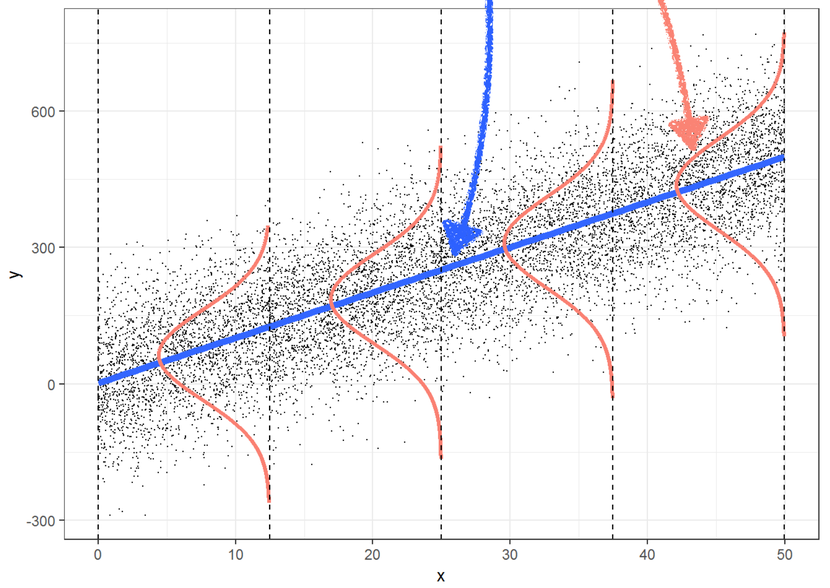

**Truncated Cauchy prior on the standard deviation**

weakly informative priors (allow for a wide range of possible values)

# Graphical representation of the model

```
1 # plotting
2 plot(m)
```



# Inference via sampling

Markov Chain Monte Carlo inference

```
1 # sampling
2 draws = mcmc(m, n_samples = 1000)
3
4 # tidy up the draws
5 df.draws = tidy_draws(draws) %>%
6   clean_names()
```

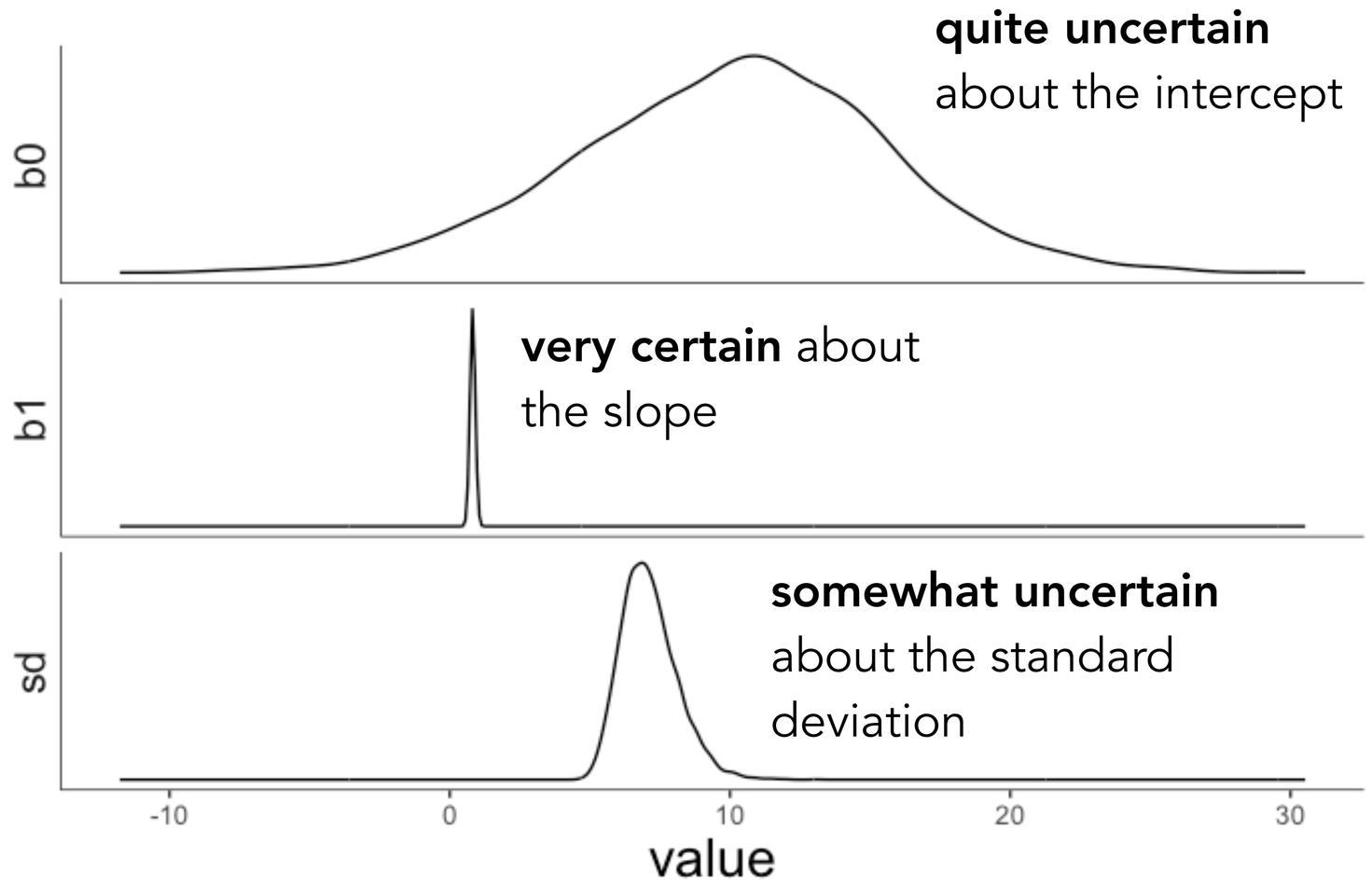
| chain | iteration | draw | b0 | b1 | sd |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 6.08 | 0.87 | 7.60 |
| 1 | 2 | 2 | 1.12 | 0.95 | 7.66 |
| 1 | 3 | 3 | -1.83 | 0.99 | 7.01 |
| 1 | 4 | 4 | -4.23 | 1.02 | 6.64 |
| 1 | 5 | 5 | 3.26 | 0.87 | 7.96 |
| 1 | 6 | 6 | -1.04 | 0.98 | 7.67 |
| 1 | 7 | 7 | -0.83 | 0.97 | 10.12 |
| 1 | 8 | 8 | -1.41 | 0.97 | 8.02 |
| 1 | 9 | 9 | 9.46 | 0.81 | 6.80 |
| 1 | 10 | 10 | 10.02 | 0.84 | 6.57 |

each of these is a solution for explaining the data

nice visualization of MCMC samplers

https://github.com/chi-feng/mcmc-demo

# Visualize the posterior



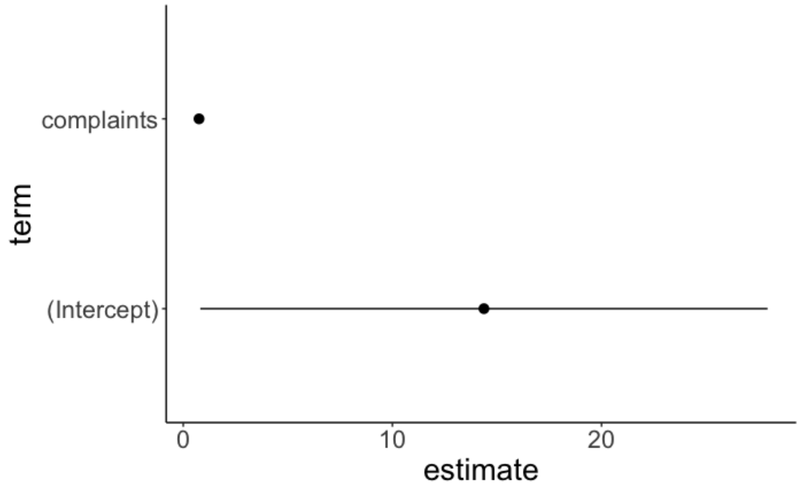

This is the solution of a Bayesian analysis.

A posterior distribution over each parameter in our model.

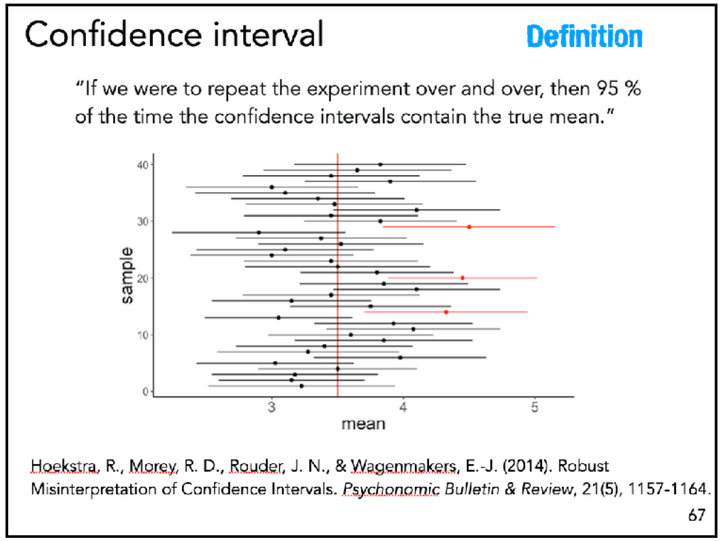
We can use this to visualize model predictions, and to test hypotheses.

# Confidence interval

```
1 fit.lm = lm(formula = rating ~ 1 + complaints,
2              data = df.attitude)
```

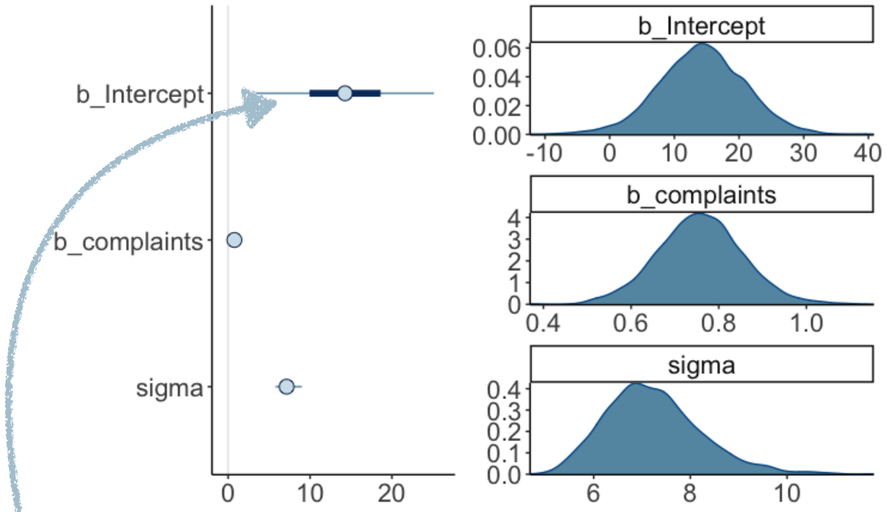



point estimate + confidence interval




# Credible interval

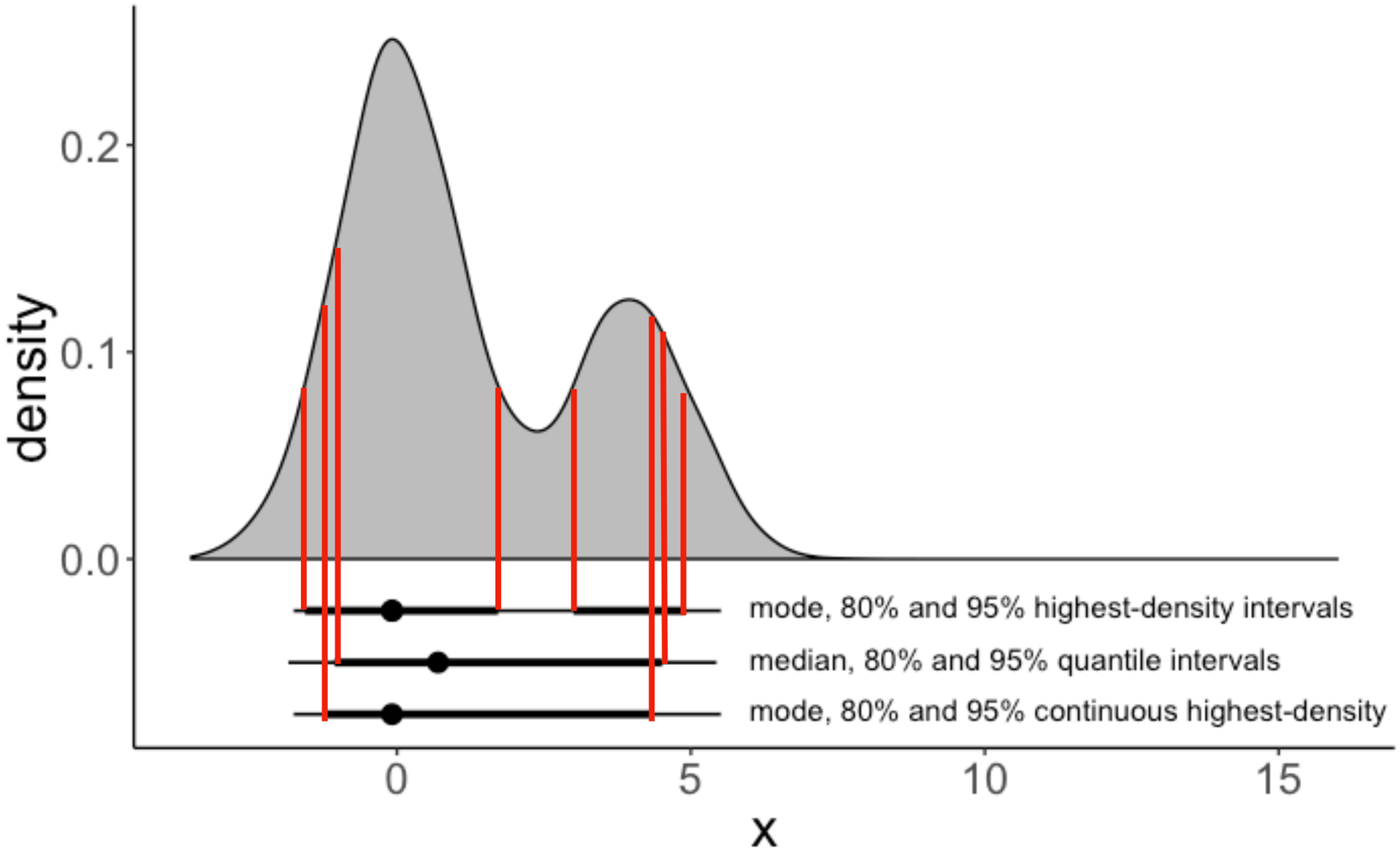
```
1 fit.brm = brm(formula = rating ~ 1 + complaints,
2               data = df.attitude)
```




full posterior distribution over each parameter

with 90% the true parameter lies within this range

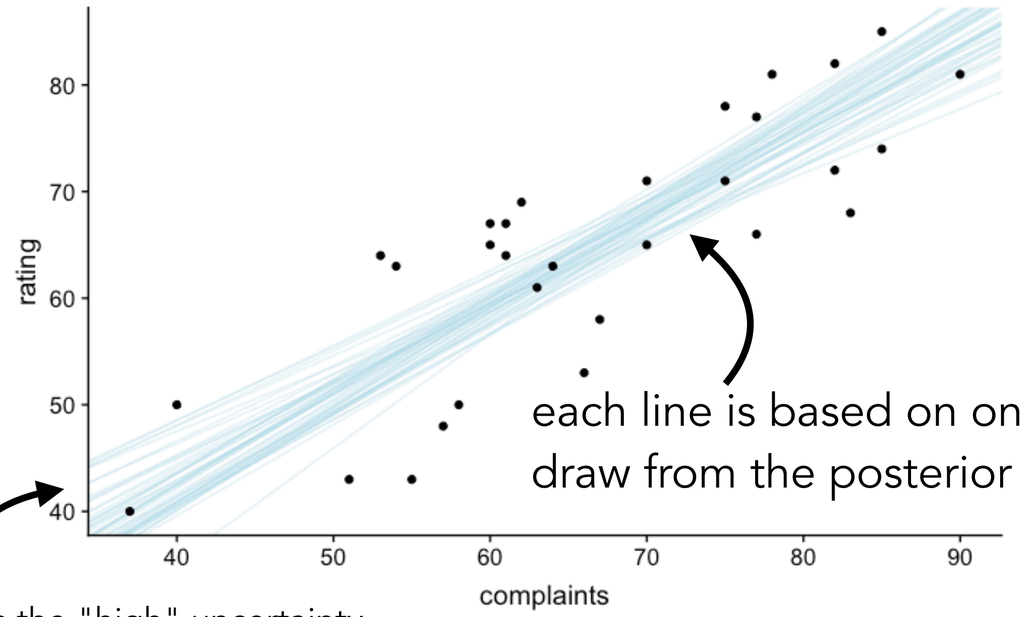
# Different kinds of credible intervals




**ways of summarizing the posterior distribution**

# Visualize the model predictions

```
ggplot(data = df.attitude,
       mapping = aes(x = complaints,
                     y = rating)) +
  geom_abline(data = df.draws %>%
                sample_n(size = 50),
              aes(intercept = b0,
                  slope = b1),
              alpha = 0.3,
              color = "lightblue") +
  geom_point()
```



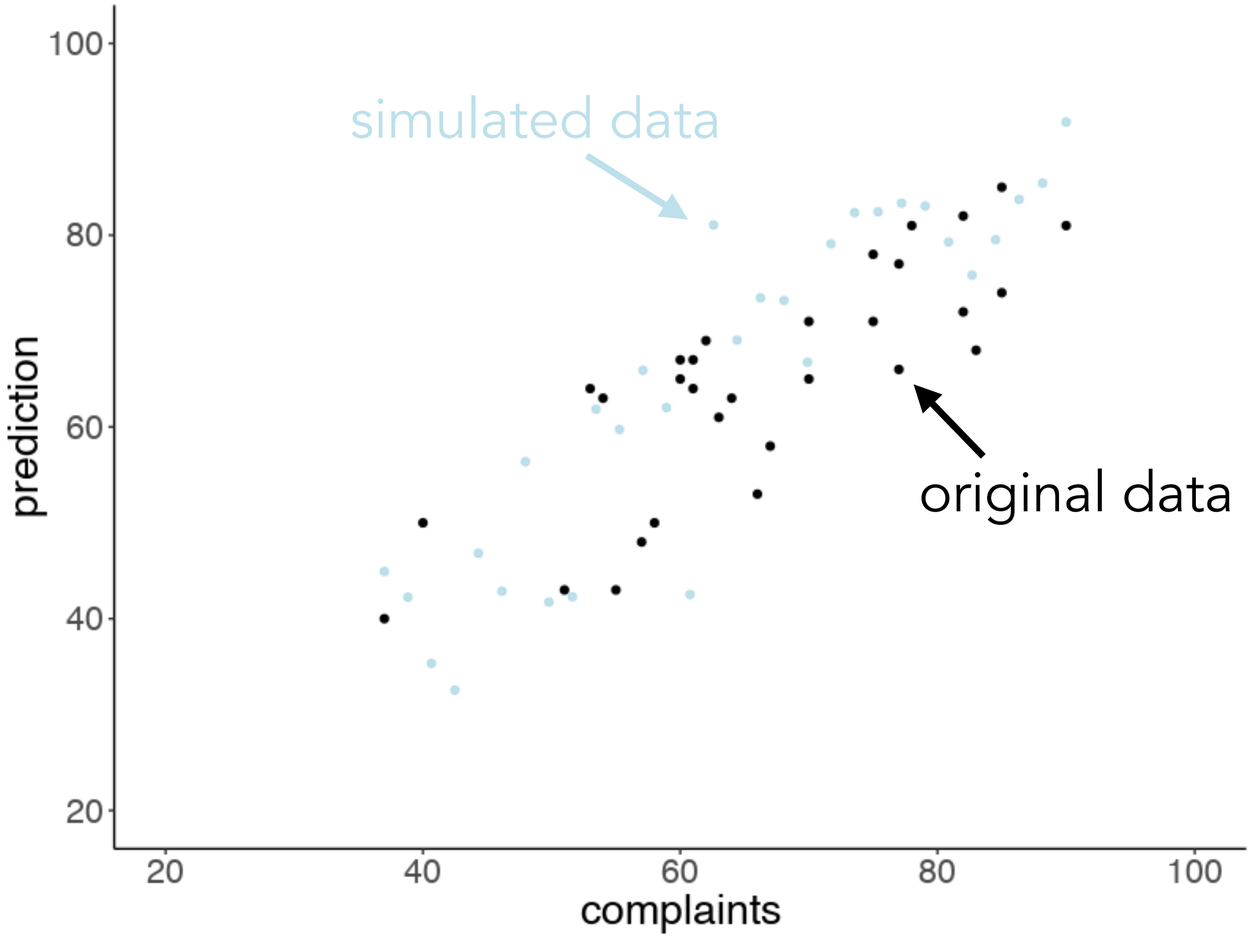

each line is based on one draw from the posterior

explains the "high" uncertainty about the intercept

| chain | iteration | draw | b0 | b1 | sd |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 6.08 | 0.87 | 7.60 |
| 1 | 2 | 2 | 1.12 | 0.95 | 7.66 |
| 1 | 3 | 3 | -1.83 | 0.99 | 7.01 |
| 1 | 4 | 4 | -4.23 | 1.02 | 6.64 |
| 1 | 5 | 5 | 3.26 | 0.87 | 7.96 |
| 1 | 6 | 6 | -1.04 | 0.98 | 7.67 |
| 1 | 7 | 7 | -0.83 | 0.97 | 10.12 |
| 1 | 8 | 8 | -1.41 | 0.97 | 8.02 |
| 1 | 9 | 9 | 9.46 | 0.81 | 6.30 |
| 1 | 10 | 10 | 10.02 | 0.84 | 6.57 |

# **Posterior** predictive check

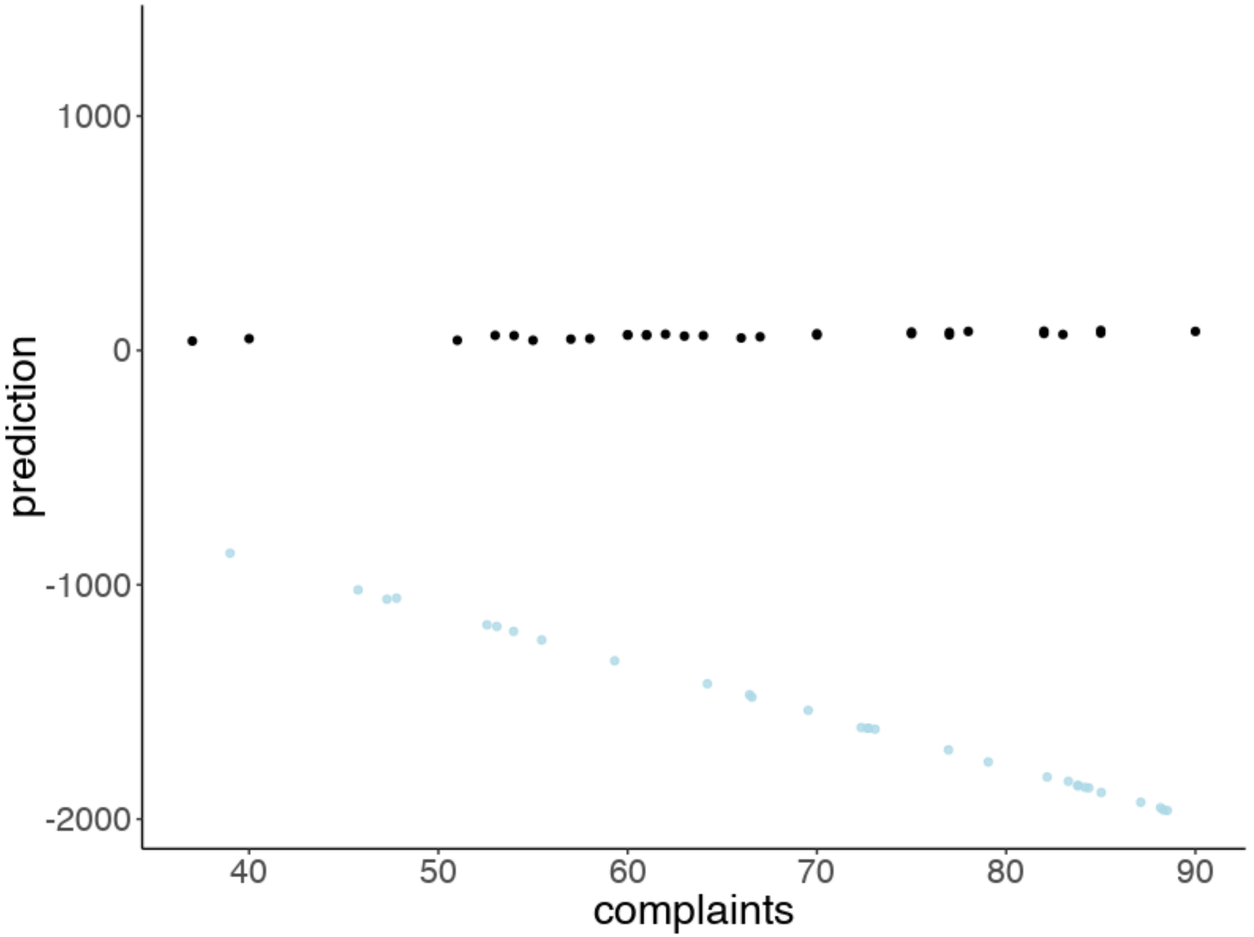
1. sample parameters from the **posterior distribution**
2. generate data using these parameters (using the likelihood function)




**Hypothetical outcome plot**

# **Prior** predictive check

1. sample parameters from the **prior distribution**
2. generate data using these parameters (using the likelihood function)




Hypothetical outcome plot

Paul Bürkner

# Doing Bayesian data analysis with BRMS

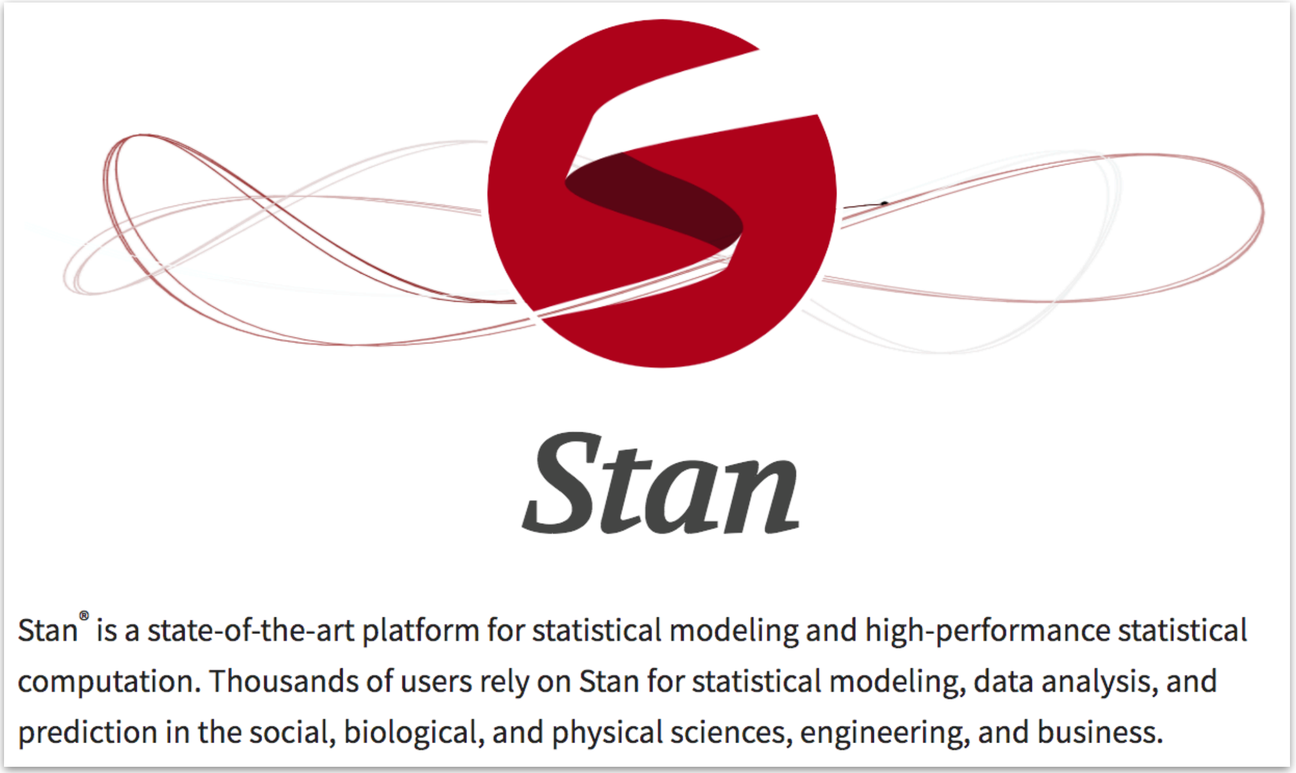
# Software packages

```
library("brms")
```


Bayesian regression modeling with Stan

- very powerful package that makes it easy to run Bayesian regression models
- we specify models using the same syntax we've already learned based on **`lm()`**, **`glm()`**, and **`lmer()`**
- `brms` turns this into Stan code and fits the model
- we can then use `tidybayes` to investigate the posterior

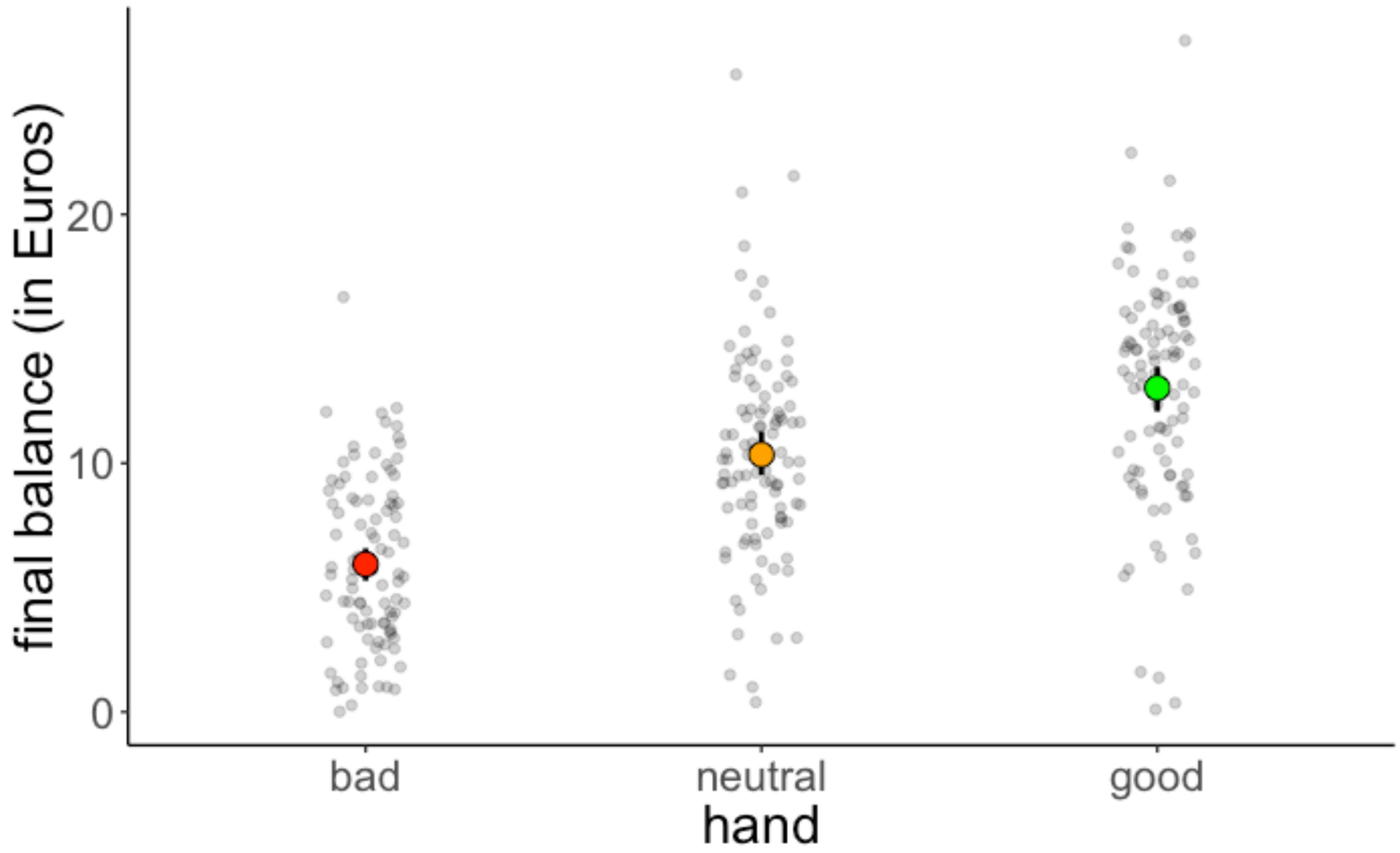
# Software packages




https://mc-stan.org/

# Poker data

# Poker data

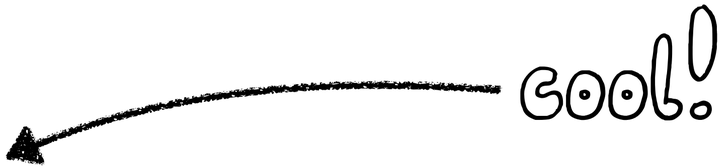


# Using `lm()`

```
1 fit.lm_poker = lm(formula = balance ~ 1 + hand,
2                   data = df.poker)
3
4 fit.lm_poker %>% summary()
```

```
Call:
lm(formula = balance ~ 1 + hand, data = df.poker)

Residuals:
     Min       1Q   Median       3Q      Max
-12.9264  -2.5902  -0.0115   2.6573  15.2834

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.9415     0.4111  14.451  < 2e-16 ***
handneutral   4.4051     0.5815   7.576 4.55e-13 ***
handgood      7.0849     0.5815  12.185  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.111 on 297 degrees of freedom
Multiple R-squared:  0.3377,   Adjusted R-squared:  0.3332
F-statistic:  75.7 on 2 and 297 DF,  p-value: < 2.2e-16
```

# Using `brm()`




```
1 fit.brm_poker = brm(formula = balance ~ 1 + hand,
2                     data = df.poker)
3
4 fit.brm_poker %>% summary()
```

```
 Family: gaussian
  Links: mu = identity; sigma = identity
Formula: balance ~ 1 + hand
   Data: df.poker (Number of observations: 300)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Population-Level Effects:
            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept       5.93      0.41     5.12     6.72 1.00     2986     2744
handneutral     4.41      0.58     3.30     5.55 1.00     3497     2903
handgood        7.10      0.58     5.99     8.29 1.00     3545     2932

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sigma     4.12      0.17     3.81     4.46 1.00     3650     2921

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

# Comparison between `lm()` and `brm()`

`lm()`

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    5.9415     0.4111  14.451  < 2e-16 ***
handneutral    4.4051     0.5815   7.576 4.55e-13 ***
handgood       7.0849     0.5815  12.185  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

`brm()`

```
Population-Level Effects:
             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept        5.93      0.41     5.12     6.72 1.00     2986     2744
handneutral      4.41      0.58     3.30     5.55 1.00     3497     2903
handgood         7.10      0.58     5.99     8.29 1.00     3545     2932
```

**almost identical results!**

# What about the priors?

```
1 fit.brm_poker = brm(formula = balance ~ 1 + hand,
2                     data = df.poker)
```

By default, `brms` uses weakly informative priors for the model parameters.

There are quite a few other defaults, let's take a look under the hood ...

# "Full" specification of the model

```
fit.brm_poker_full = brm(
  formula = balance ~ 1 + hand,
  family = "gaussian",
  data = df.poker,
  prior = c(
    prior(normal(0, 10), class = "b", coef = "handgood"),
    prior(normal(0, 10), class = "b", coef = "handneutral"),
    prior(student_t(3, 3, 10), class = "Intercept"),
    prior(student_t(3, 0, 10), class = "sigma")
  ),
  inits = list(
    list(Intercept = 0, sigma = 1, handgood = 5, handneutral = 5),
    list(Intercept = -5, sigma = 3, handgood = 2, handneutral = 2),
    list(Intercept = 2, sigma = 1, handgood = -1, handneutral = 1),
    list(Intercept = 1, sigma = 2, handgood = 2, handneutral = -2)
  ),
  iter = 4000,
  warmup = 1000,
  chains = 4,
  file = "cache/brm_poker_full",
  seed = 1
)
```

likelihood

priors

initialization

how many runs in the inference chain

how long for the warmup

how many chains

save the model result

make reproducible

fitting Bayesian models takes some time, so storing results is key

# Turned into Stan code

```
// generated with brms 2.7.0
functions {
}
data {
  int<lower=1> N;  // total number of observations
  vector[N] Y;  // response variable
  int<lower=1> K;  // number of population-level effects
  matrix[N, K] X;  // population-level design matrix
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  int Kc = K - 1;
  matrix[N, K - 1] Xc;  // centered version of X
  vector[K - 1] means_X;  // column means of X before centering
  for (i in 2:K) {
    means_X[i - 1] = mean(X[, i]);
    Xc[, i - 1] = X[, i] - means_X[i - 1];
  }
}
parameters {
  vector[Kc] b;  // population-level effects
  real temp_Intercept;  // temporary intercept
  real<lower=0> sigma;  // residual SD
}
transformed parameters {
}
model {
  vector[N] mu = temp_Intercept + Xc * b;
  // priors including all constants
  target += normal_lpdf(b[1] | 0, 10);
  target += normal_lpdf(b[2] | 0, 10);
  target += student_t_lpdf(temp_Intercept | 3, 3, 10);
  target += student_t_lpdf(sigma | 3, 0, 10)
    - 1 * student_t_lccdf(0 | 3, 0, 10);
  // likelihood including all constants
  if (!prior_only) {
    target += normal_lpdf(Y | mu, sigma);
  }
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = temp_Intercept - dot_product(means_X, b);
}
```

- probabilistic programming language
- flexible construction of Bayesian models
- ports have been written for R, Python, Julia, ...
- implements a fast inference algorithm

# Results

## posterior samples

| b_Intercept | b_handneutral | b_handgood | sigma |
|---|---|---|---|
| 5.97 | 4.27 | 7.48 | 3.94 |
| 5.11 | 5.25 | 7.40 | 3.91 |
| 7.03 | 3.78 | 5.80 | 4.48 |
| 5.72 | 4.18 | 7.25 | 4.00 |
| 6.01 | 4.44 | 6.15 | 4.57 |
| 5.94 | 4.69 | 6.72 | 4.36 |
| 6.39 | 3.84 | 6.40 | 3.92 |
| 5.24 | 5.15 | 7.69 | 4.16 |
| 6.12 | 4.51 | 7.20 | 4.14 |
| 6.43 | 3.71 | 6.37 | 4.13 |
| 5.85 | 5.01 | 7.32 | 4.00 |
| 6.51 | 3.58 | 6.62 | 3.95 |
| 5.85 | 4.45 | 7.62 | 4.17 |
| 5.80 | 5.45 | 6.36 | 4.10 |
| 5.48 | 5.51 | 7.22 | 3.99 |

⋮

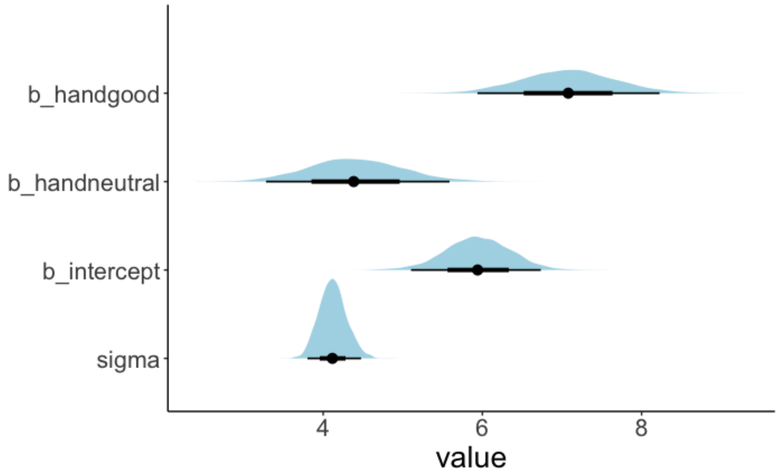
maximum a posteriori

## summary of posterior

| parameter | lower | mode | upper |
|---|---|---|---|
| b_handgood | 5.97 | 7.07 | 8.27 |
| b_handneutral | 3.21 | 4.43 | 5.51 |
| b_intercept | 5.17 | 5.95 | 6.77 |
| sigma | 3.81 | 4.12 | 4.47 |

MAP estimate and 95% highest density interval

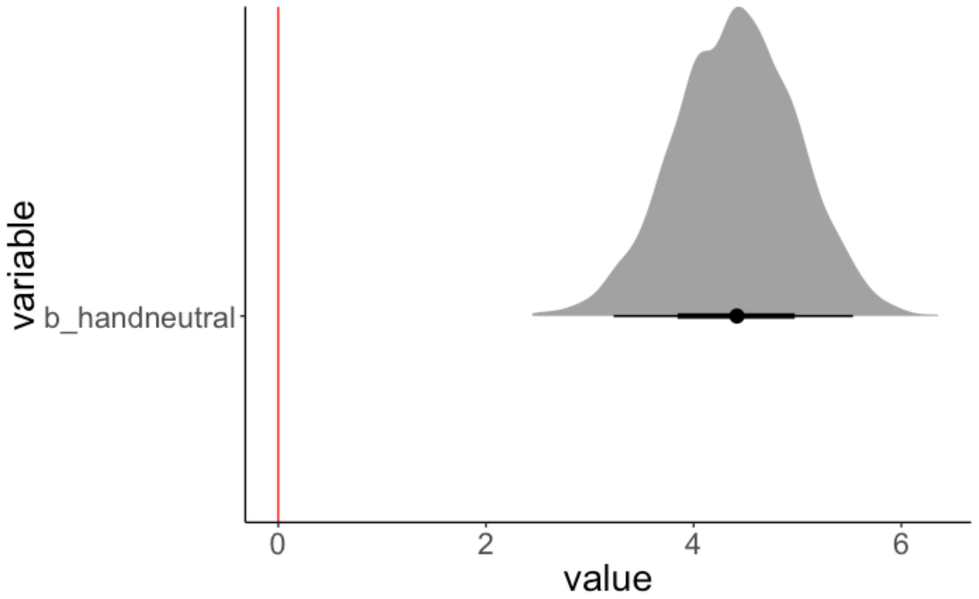
## visualization



# Testing hypotheses

# Asking questions based on the posterior

**Do neutral hands earn more money than bad hands?**



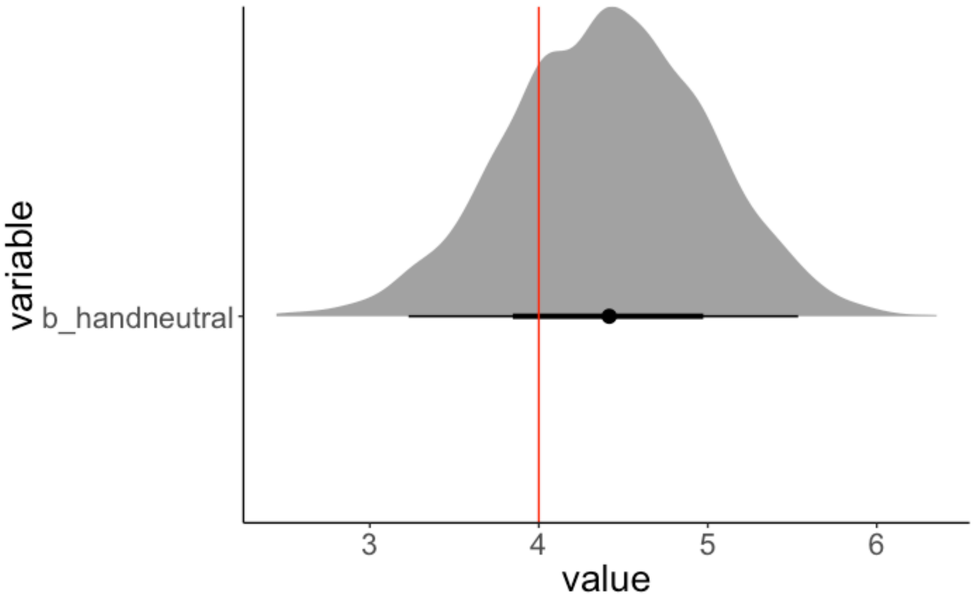

What's the probability that handneutral is less than 0?

```
1 hypothesis(fit.brm,
2            hypothesis = "handneutral < 0")
```

p = 0

# Asking questions based on the posterior

**Do neutral hands earn much more money than bad hands?**



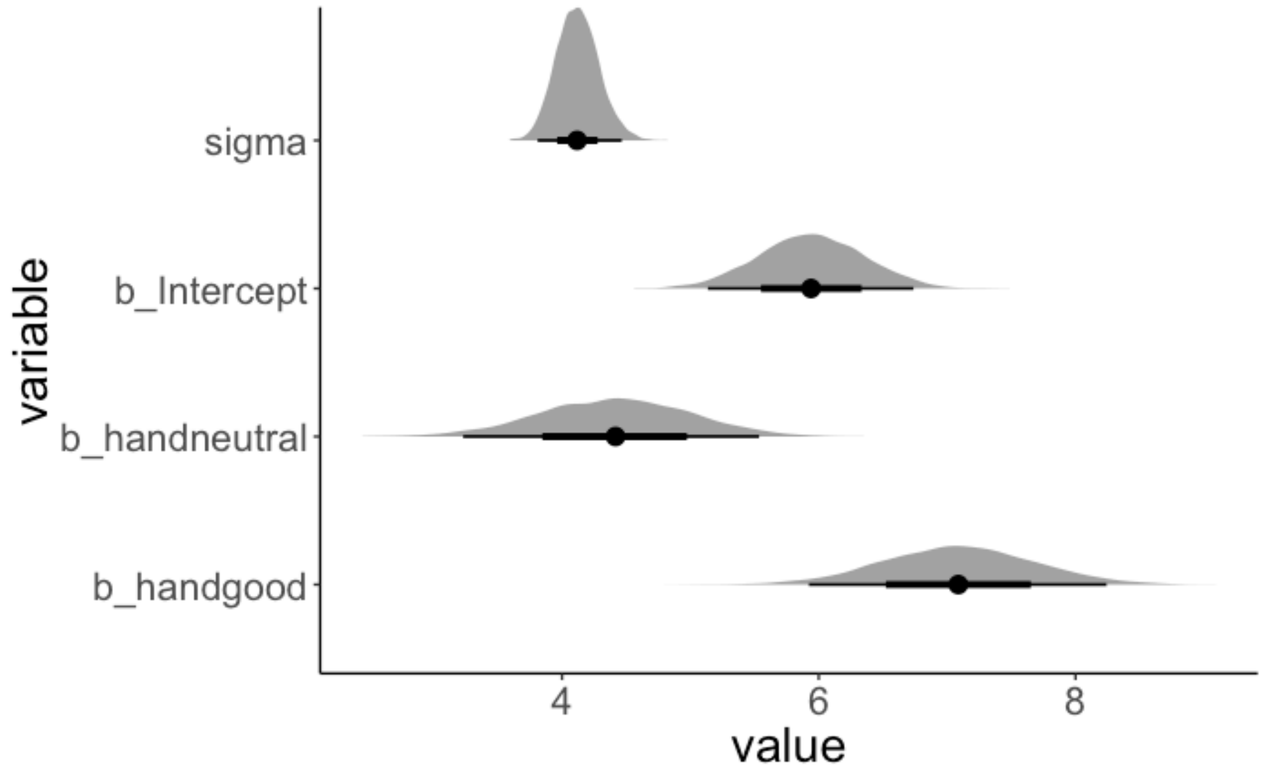

What's the probability that the difference between neutral and bad hands is **more than 4**?

```
1 hypothesis(fit.brm,
2            hypothesis = "handneutral > 4")
```

p = 0.75

# Asking questions based on the posterior

**Do good hands make twice as much as bad hands?**

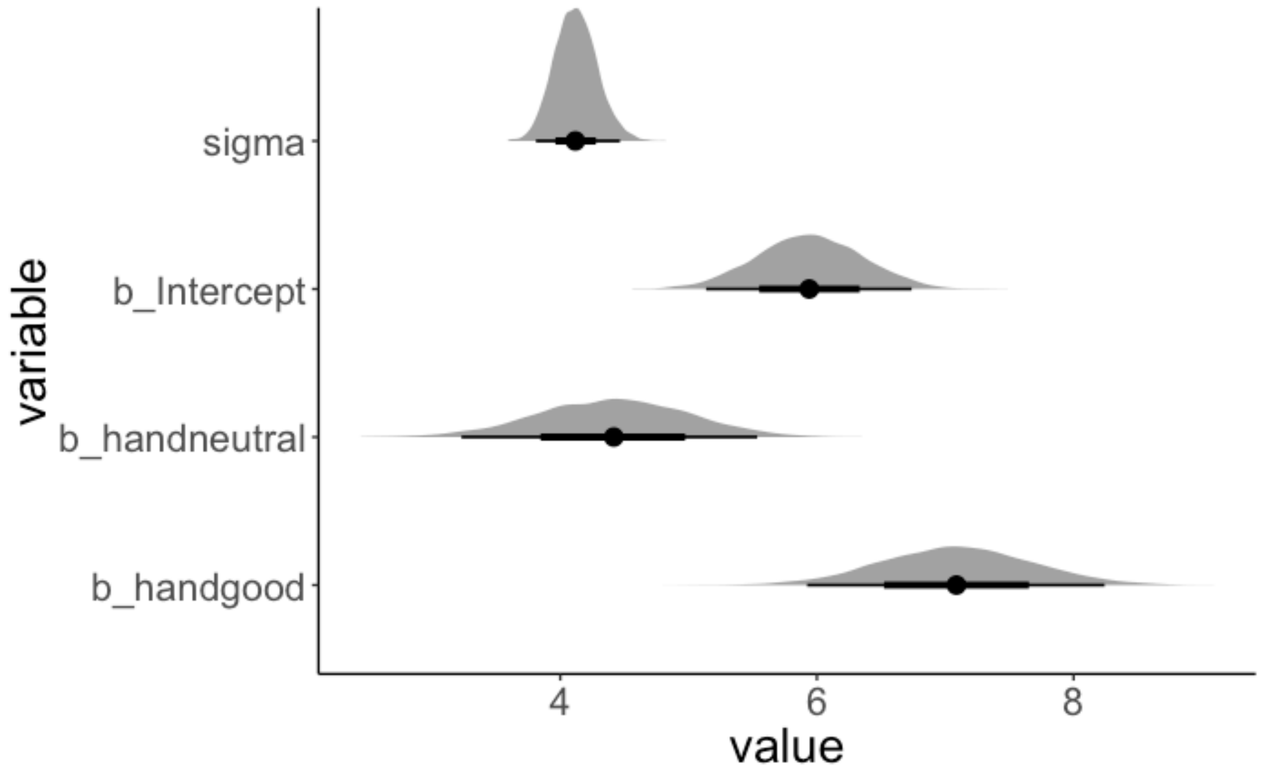



```
1 hypothesis(fit.brm,
2              hypothesis = "handgood + Intercept > 2 * Intercept")
```

p = 0.89

# Asking questions based on the posterior

**Are neutral hands worse than bad and good hands combined?**




```
1 hypothesis(fit.brm,
2              hypothesis = "Intercept + handneutral < (Intercept + Intercept + handgood) / 2")
```

p = 0.04

# Testing hypothesis

```
df.hypothesis = fit.brm %>%
  posterior_samples() %>%
  clean_names() %>%
  select(starts_with("b_")) %>%
  mutate(neutral = b_intercept + b_handneutral,
         bad_good_average = (b_intercept + b_intercept + b_handgood)/2,
         hypothesis = neutral < bad_good_average)
```

samples from the posterior

| b_intercept | b_handneutral | b_handgood | neutral | bad_good_average | hypothesis |
|---|---|---|---|---|---|
| 6.07 | 4.10 | 7.20 | 10.17 | 9.67 | FALSE |
| 6.06 | 4.44 | 6.95 | 10.49 | 9.53 | FALSE |
| 5.88 | 5.00 | 6.73 | 10.87 | 9.24 | FALSE |
| 5.85 | 4.78 | 6.18 | 10.63 | 8.94 | FALSE |
| 5.86 | 4.46 | 7.68 | 10.32 | 9.70 | FALSE |

```
df.hypothesis %>%
  summarize(p = sum(hypothesis)/n())
```

p = 0.04

# Testing hypotheses

Having a posterior distribution allows us to ask questions about the data in a very flexible way!

# The "emmeans" package is your friend!

```
1 fit.brm_poker %>%
2   emmeans(specs = consec ~ hand)
```

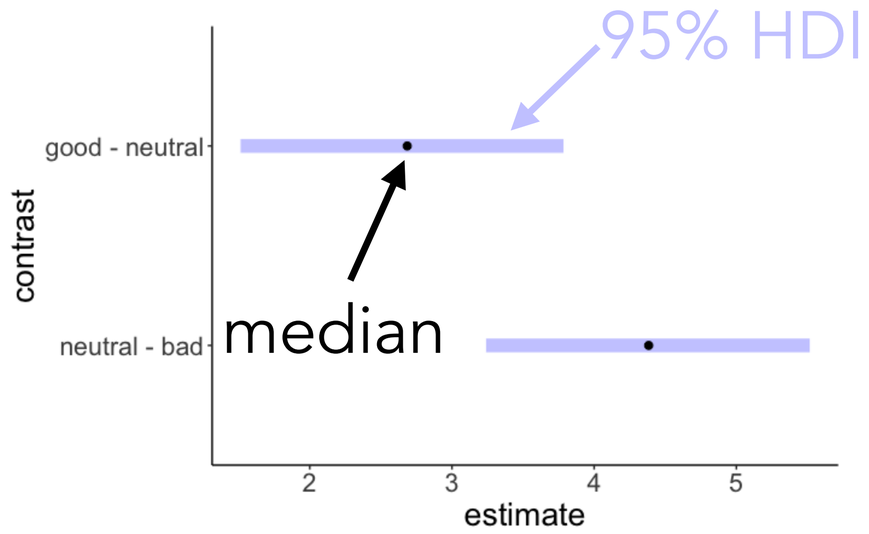
estimated mean for each group →

```
$emmeans
 hand    emmean lower.HPD upper.HPD
 bad       5.94      5.16      6.78
 neutral  10.34      9.55     11.15
 good     13.02     12.22     13.82

Point estimate displayed: median
HPD interval probability: 0.95

$contrasts
 contrast       estimate lower.HPD upper.HPD
 neutral - bad      4.38      3.24      5.52
 good - neutral     2.69      1.51      3.78

Point estimate displayed: median
HPD interval probability: 0.95
```

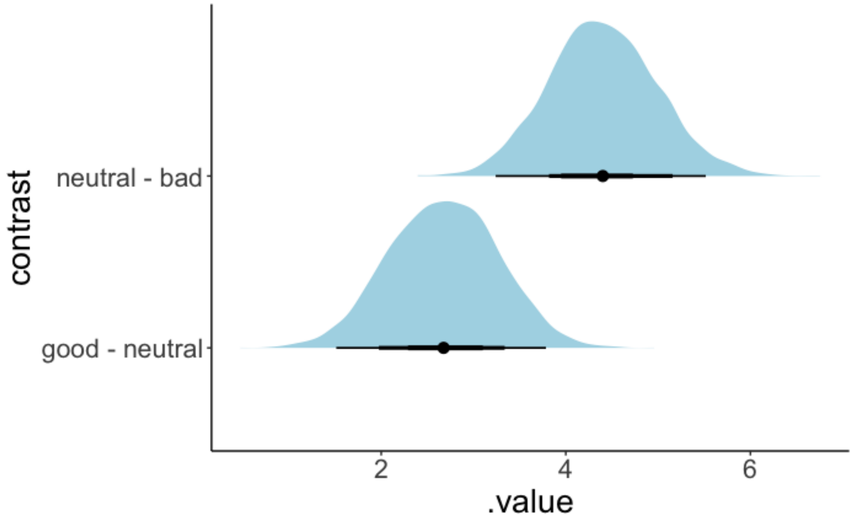
contrasts →

# Visualizing the contrasts

```
fit.brm_poker %>%
  emmeans(specs = consec ~ hand) %>%
  pluck("contrasts") %>%
  plot()
```




```
fit.brm_poker %>%
  emmeans(specs = consec ~ hand) %>%
  pluck("contrasts") %>%
  gather_emmeans_draws() %>%
  ggplot(mapping = aes(y = contrast,
                       x = .value)) +
  stat_halfeye(fill = "lightblue",
               point_interval = mean_hdi,
               .width = c(0.5, 0.75, 0.95))
```



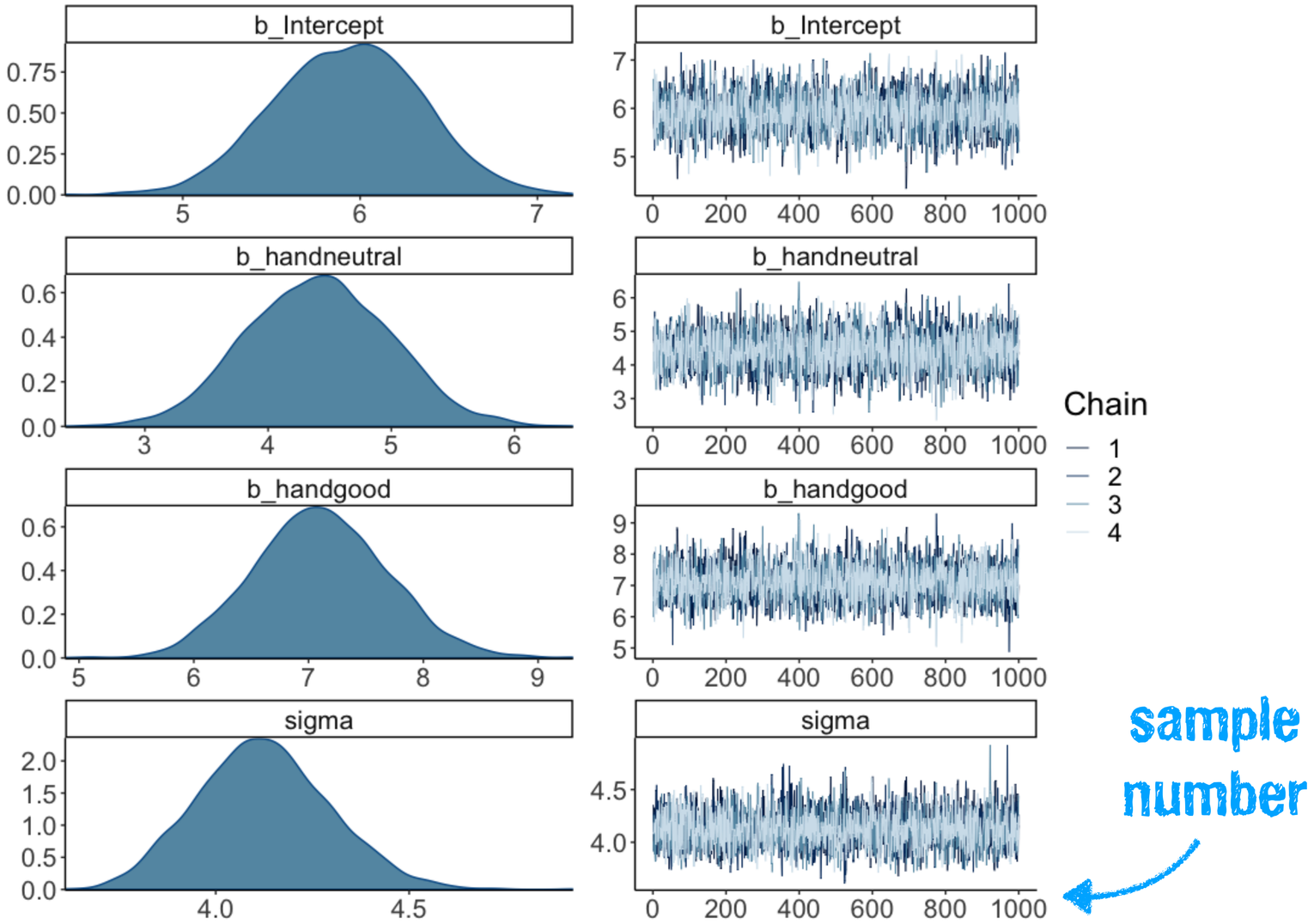

mean, 50% HDI, 75% HDI, 95% HDI

# Model evaluation

# 1. Check whether inference worked

# Can we trust the model results?

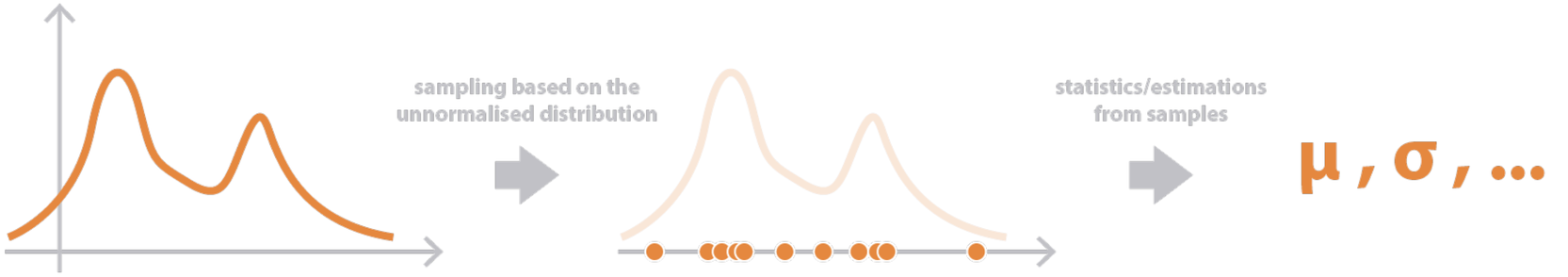
```
plot(fit.brm_poker)
```



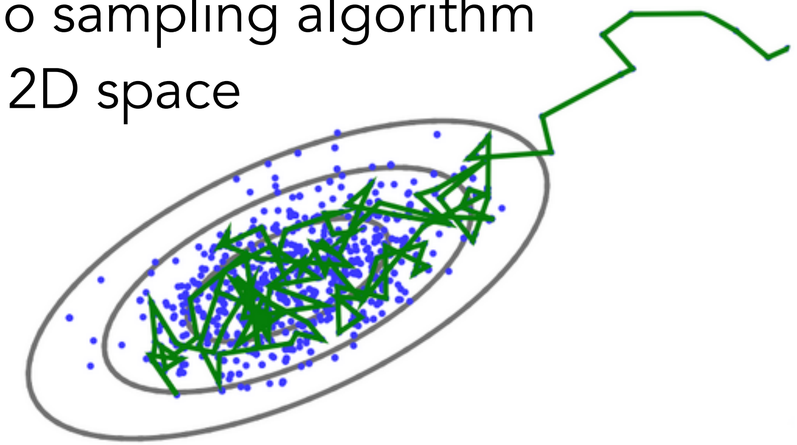

https://chi-feng.github.io/mcmc-demo/

# Can we trust the model results?

## Inference via Markov Chain Monte Carlo (MCMC)



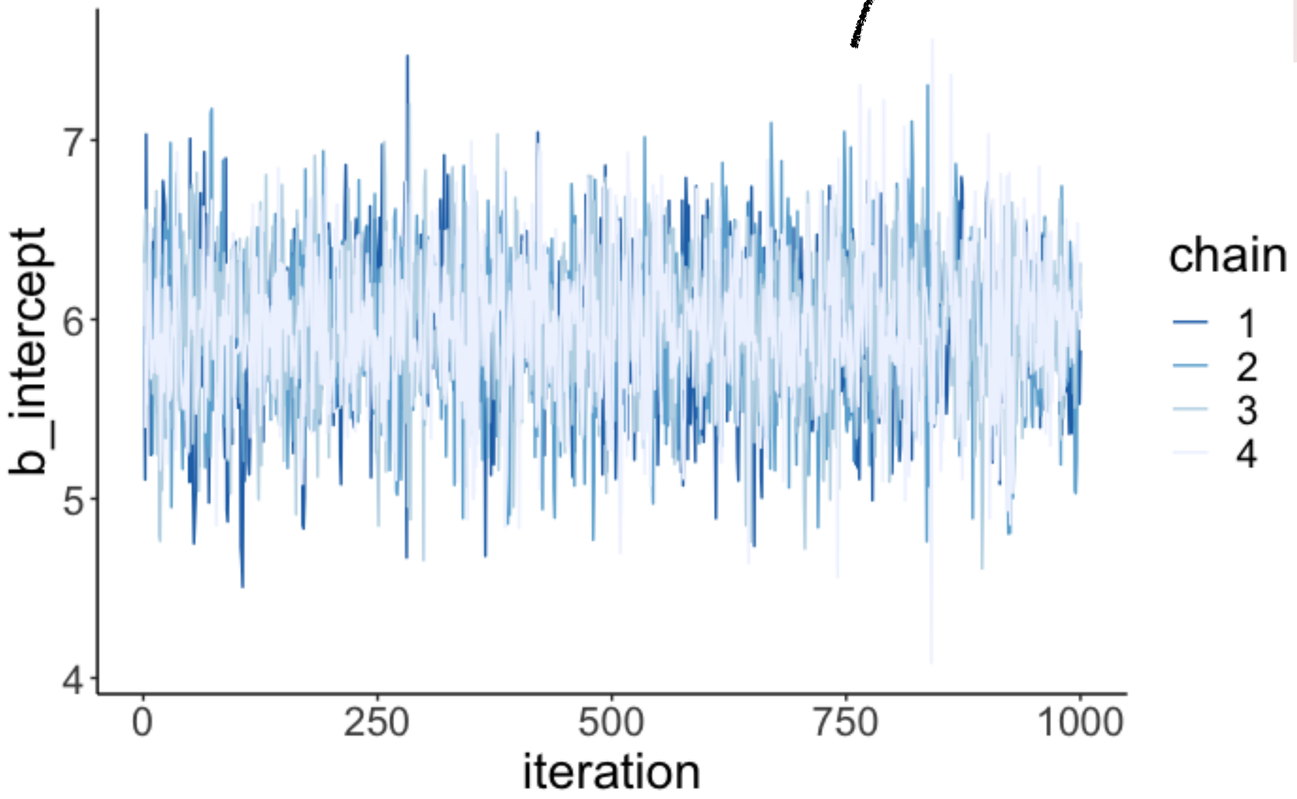

Markov Chain Monte Carlo sampling algorithm in a 2D space



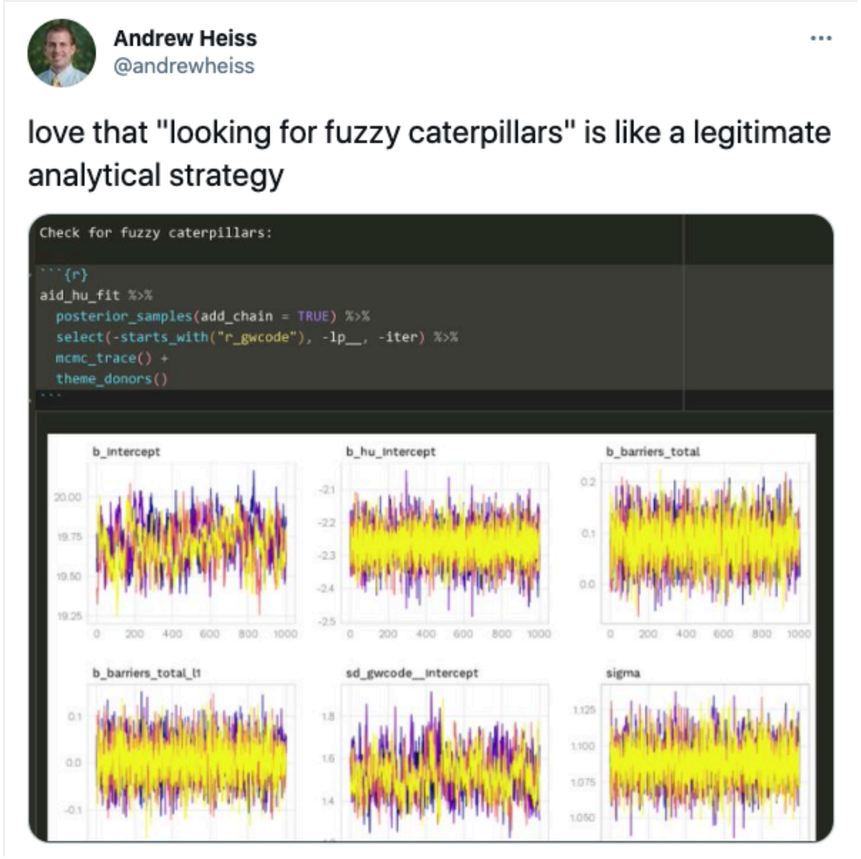
**goal**: draw **independent** samples from the posterior distribution

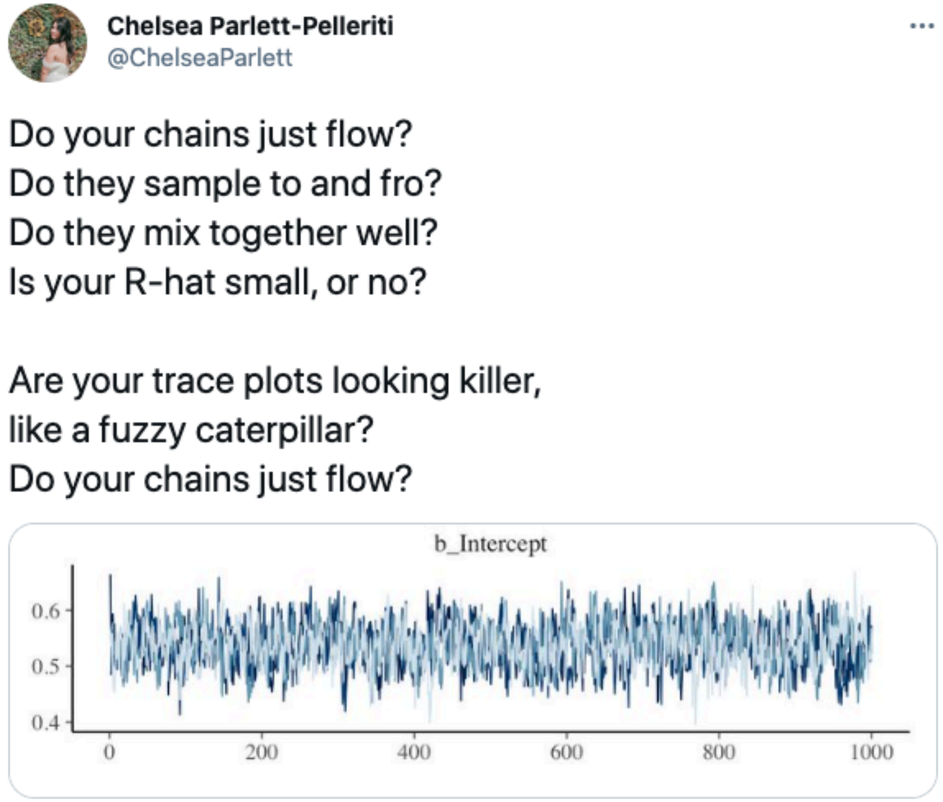
# Can we trust the model results?

looks like a fuzzy caterpillar



# Stats twitter chimes in ...



# When things don't work out

only two data points!

incredibly wide uniform priors

10000000000

```
1  df.data = tibble(y = c(-1, 1))
2
3  fit.brm_wrong = brm(data = df.data,
4                      family = gaussian,
5                      formula = y ~ 1,
6                      prior = c(prior(uniform(-1e10, 1e10), class = Intercept),
7                                prior(uniform(0, 1e10), class = sigma)),
8                      inits = list(list(Intercept = 0, sigma = 1),
9                                   list(Intercept = 0, sigma = 1)),
10                     iter = 4000,
11                     warmup = 1000,
12                     chains = 2,
13                     file = "cache/brm_wrong")
```

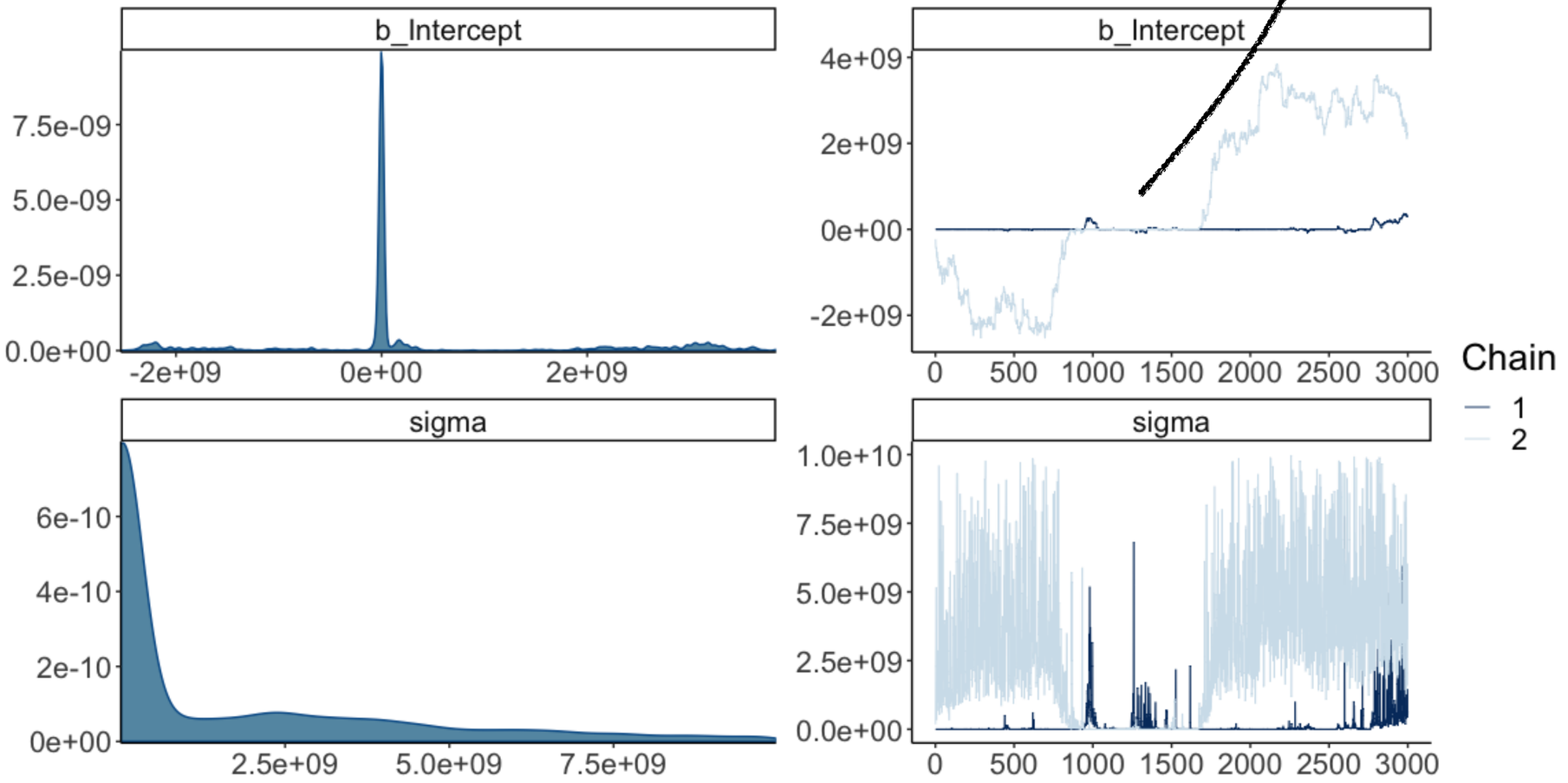
# When things don't work out

```
summary(fit.brm_wrong)
```

```
The model has not converged (some Rhats are > 1.1). Do not analyse the results!
We recommend running more iterations and/or setting stronger priors.There were 1203 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: gaussian
  Links: mu = identity; sigma = identity
Formula: y ~ 1
   Data: df.data (Number of observations: 2)
Samples: 2 chains, each with iter = 4000; warmup = 1000; thin = 1;
         total post-warmup samples = 6000

Population-Level Effects:
             Estimate     Est.Error       l-95% CI      u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept 357550121.58 1416057299.71 -2244033111.47 3333594132.43 1.78        3       24

Family Specific Parameters:
           Estimate     Est.Error l-95% CI      u-95% CI Rhat Bulk_ESS Tail_ESS
sigma 1524412740.64 2392424321.98 21668.93 8317582240.06 1.40        4       41

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

# When things don't work out

doesn't look like a fuzzy caterpillar



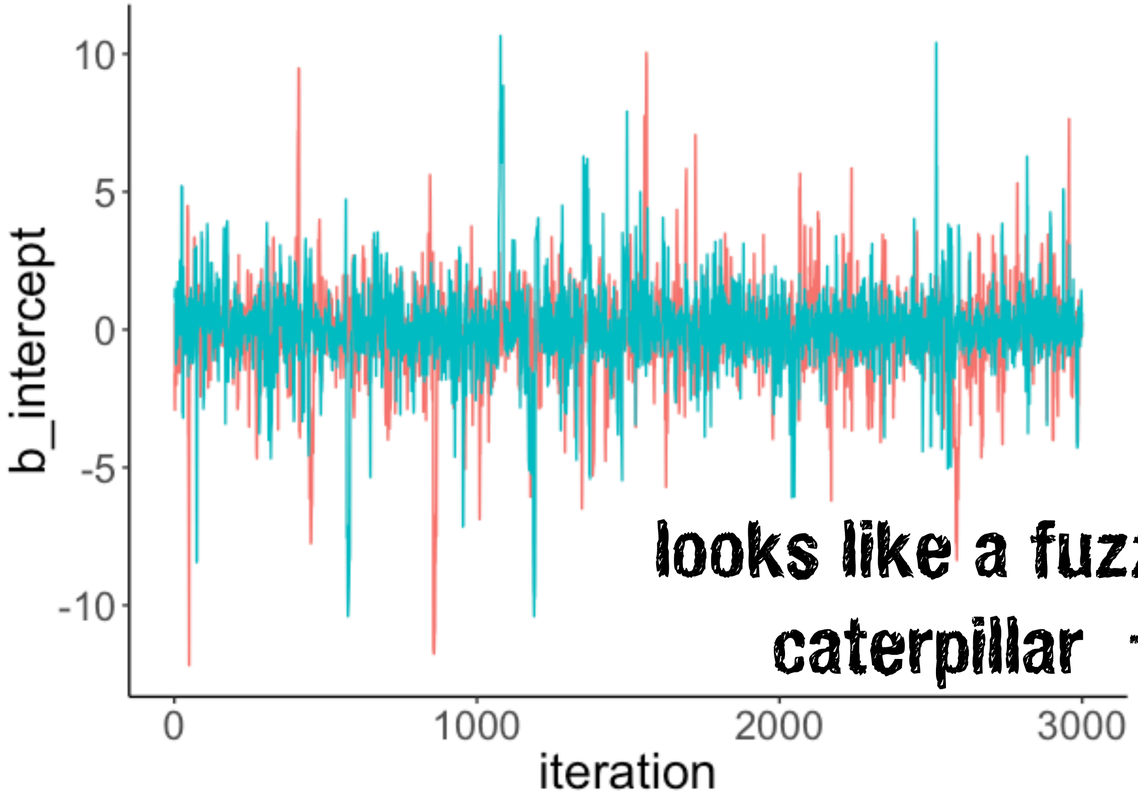
# Having somewhat informative priors fixes things

```
1  fit.brm_right = brm(data = df.data,
2                      family = gaussian,
3                      formula = y ~ 1,
4                      prior = c(prior(normal(0, 10), class = Intercept), # more reasonable priors
5                                prior(cauchy(0, 1), class = sigma)),
6                      iter = 4000,
7                      warmup = 1000,
8                      chains = 2,
9                      seed = 1,
10                     file = "cache/brm_right")
```

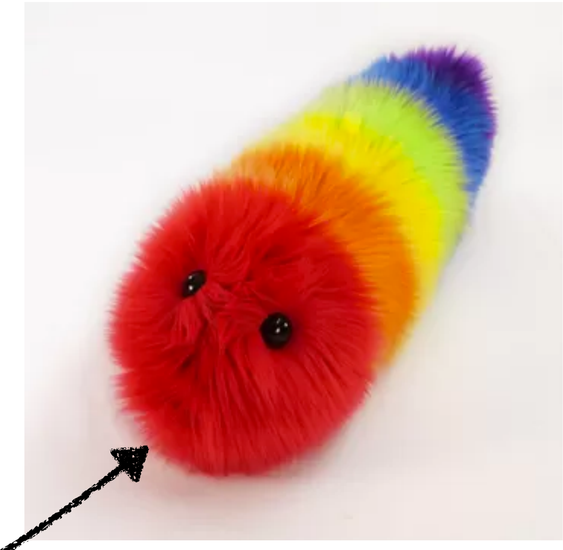
more reasonable priors

```
Family: gaussian
  Links: mu = identity; sigma = identity
Formula: y ~ 1
   Data: list(y = c(-1, 1)) (Number of observations: 2)
Samples: 2 chains, each with iter = 4000; warmup = 1000; thin = 1;
         total post-warmup samples = 6000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept    -0.06      1.72    -3.78     3.27       1033 1.00

Family Specific Parameters:
      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sigma     2.21      6.99     0.61     6.92       1006 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

# Having somewhat informative priors fixes things




looks like a fuzzy caterpillar



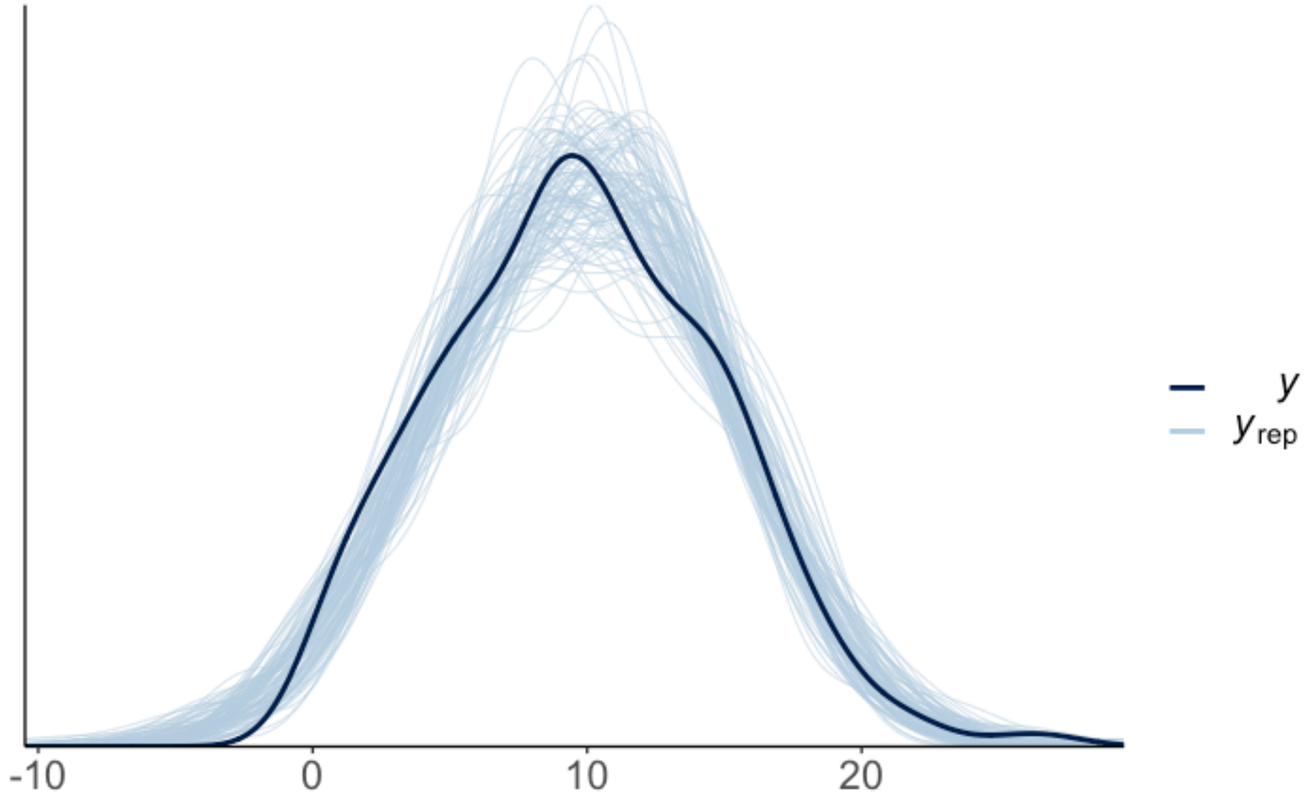
**if things go wrong**:

- set more informative priors
- run more warm-up samples
- adjust the sampling algorithm as suggested via the `control` argument

# 2. Visualize model predictions

# Posterior predictive check

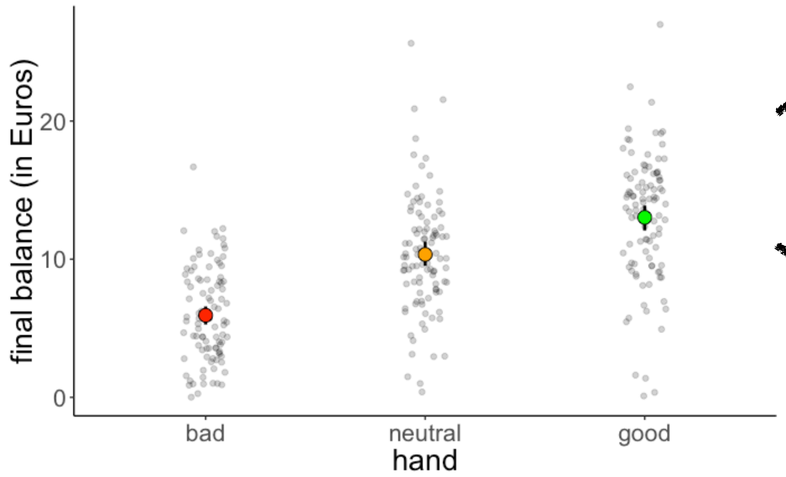
```
pp_check(fit.brm, nsamples = 100)
```



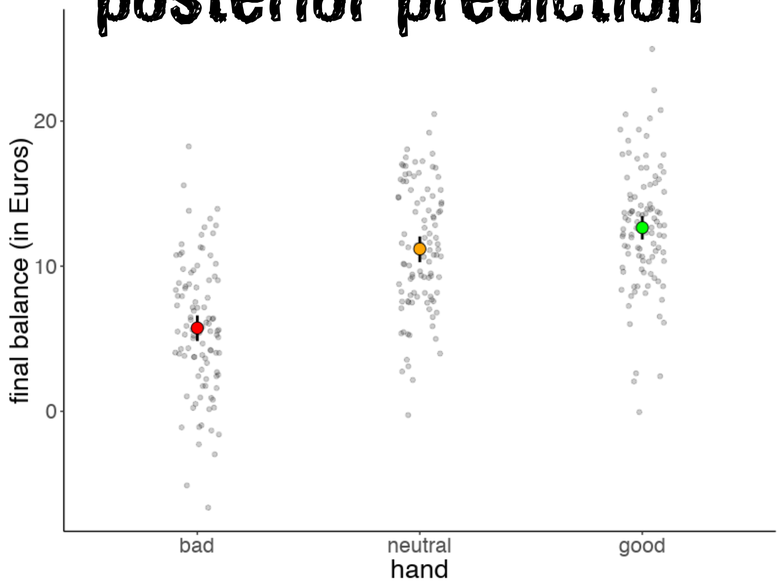

**The model accurately captures the distribution of the response variable**

# Posterior predictive check

**original data**




**posterior prediction**



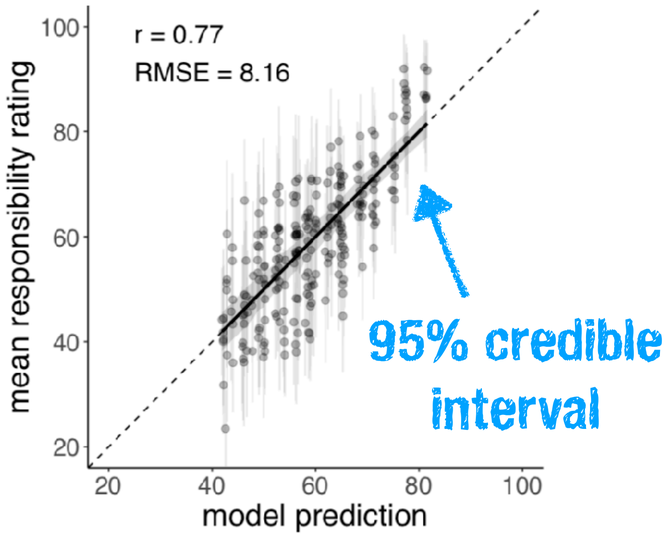

**prior prediction**


**take a look at course notes for how to make these**

# Reporting results

# Reporting results

## Plots




## Tables

Table 1

*Estimates of the mean, standard error, and 95% HDIs of the different predictors in the Bayesian mixed effects model. Note:* `n_causes` *= number of causes.*

```
responsibility ~ 1 + surprise + pivotality + n_causes + (1 + surprise + pivotality + n_causes | participant)
```

| term | estimate | std.error | lower 95% HDI | upper 95% HDI |
|---|---|---|---|---|
| intercept | 59.94 | 3.25 | 54.70 | 65.22 |
| surprise | 21.68 | 4.57 | 14.17 | 29.23 |
| pivotality | 13.52 | 1.82 | 10.47 | 16.53 |
| n_causes | -5.72 | 0.50 | -6.55 | -4.90 |

model formula

parameter estimates

## Text

We computed a Bayesian mixed effects model with random intercepts and slopes to predict participants' responsibility judgments (see Table 1). Figure 6b shows a scatter plot of the model predictions and participants' responsibility judgments for the full set of 170 scenarios (with 250 judgments). Overall, the model predicts participants' responsibility judgments well with r = .77 and RMSE = 8.16. Table 1 shows the estimates of the different predictors. As can be seen, none of the predictors' 95% HDIs overlap with 0.[1]

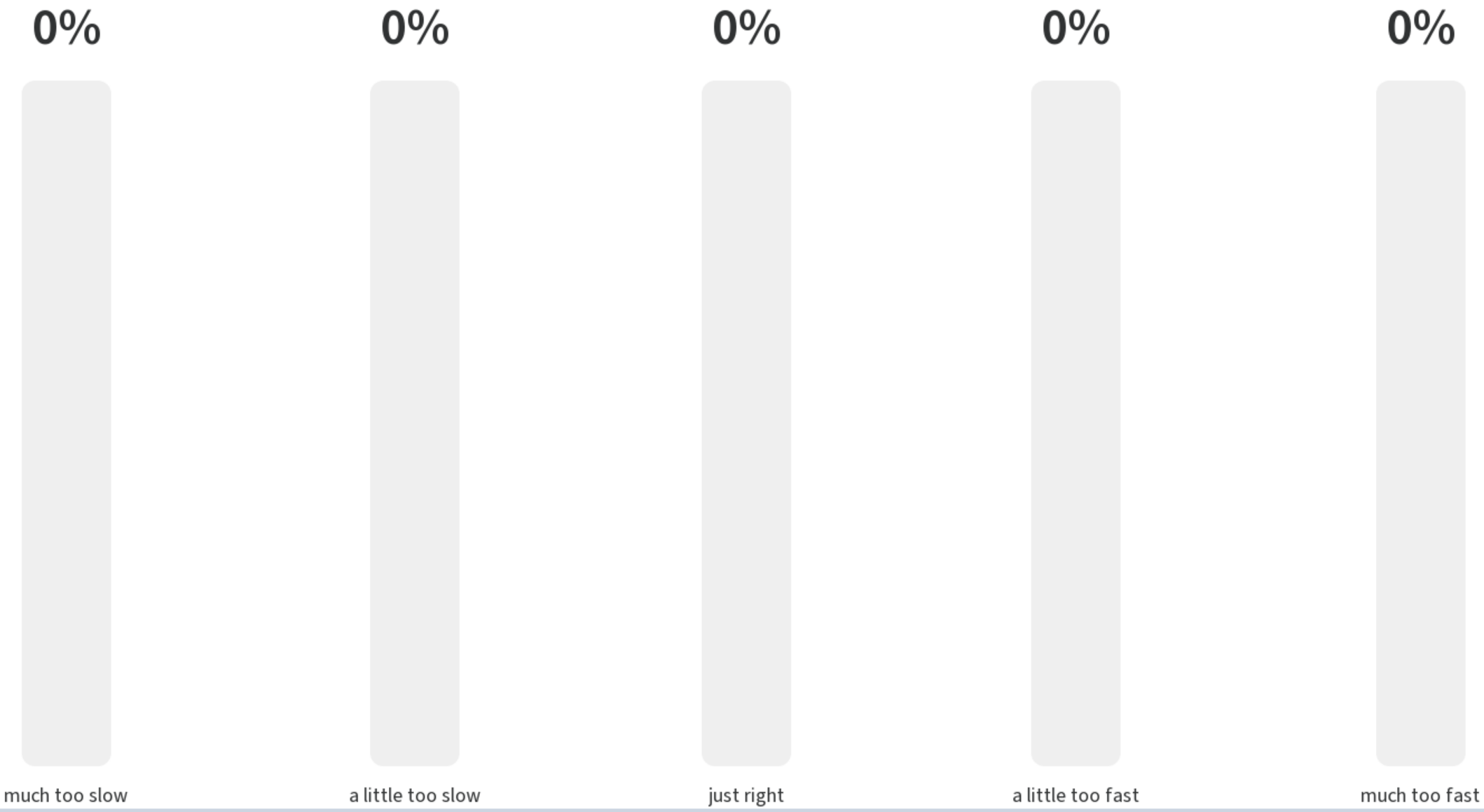
[1]For any statistical claim, we report the mean of the posterior distribution together with the 95% highest-density interval (HDI). All Bayesian models were written in Stan (Carpenter et al., 2017) and accessed with the brms package (Bürkner, 2017) in R (R Core Team, 2019).

# Summary

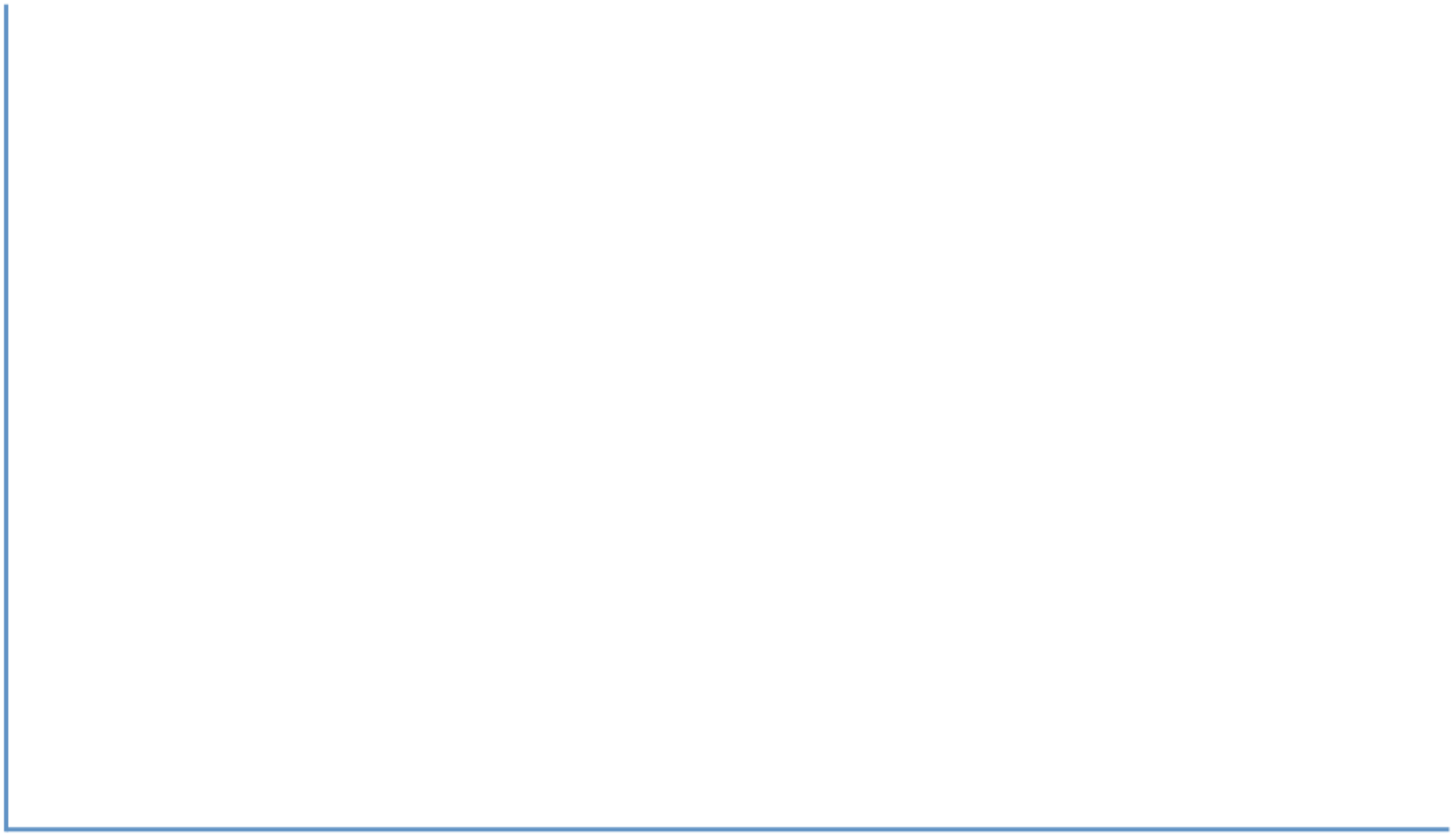
- Quick recap
- Doing Bayesian data analysis with greta
- Doing Bayesian data analysis **with BRMS**
  - Testing hypotheses
  - Model evaluation

# Feedback N

0



# How was the pace of today's class?



# Thank you!