Your First Model!

All statistical analyses are based on models

Tim Frasier

First, Note on Tests
vs
Generalized Linear Models

Tests vs Generalized Linear Model

- In stats courses (or elsewhere), you likely learned different tests
  - Each applicable in different situations

***t*-test**

ANOVA

ANCOVA

Binomial Test

Linear Regression

Wilcoxon Rank-Sum Test

Chi-Square Test

Fisher's Exact Test

etc.

Tests vs Generalized Linear Model

- In stats courses (or elsewhere), you likely learned different tests
  - Each applicable in different situations
- Often, these are taught and/or learned as independent things
  - Underlying principles not clear
  - Memorization rather than understanding

A Secret

- They’re all variations on a generalized linear model
    - One set of underlying principles, different types of parameters

A Secret

- They’re all variations on a generalized linear model
  - One set of underlying principles, different types of parameters
  - Once you understand basic principles, no need to memorize anything, just build an appropriate model!

Back to First Model

Data

Data

- `calving.csv`

- Reproductive data for North Atlantic right whales (5 years, 2010-2014)
  - 1 if yes, 0 if no

```
0 0 1 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 1...
```

Data

- Females are capable of giving birth once every 3 years

- Average inter-birth interval is ~6 years

Data

- Load appropriate libraries

```
library(ggplot2)
library(rstan)
```

- Read the data into R

```
calving = read.table("calving.csv", header = FALSE, sep = ",")
```

Data

- By using the `str` function, we can see that R has formatted the data as a data frame with a single variable (and labeled it "`V1`")

```
str(calving)

'data.frame':	218 obs. of  1 variable:
 $ V1: int  0 0 1 0 1 0 0 0 0 0 ...
```

Data

- First let's get a feel for the data

A very important thing to always do first

Data

- How many females are in this data set?

```
females = length(calving$V1)
females

[1] 218
```

Data

- How many females are in this data set?

```
females = length(calving$V1)
females

[1] 218
```

- How many females gave birth?

```
mothers = sum(calving$V1)
mothers

[1] 79
```

Data

- How many females are in this data set?

```
females = length(calving$V1)
females

[1] 218
```

- How many females gave birth?

```
mothers = sum(calving$V1)
mothers

[1] 79
```

- What percentage of the females is this?

```
mothers / females

[1] 0.3623853
```

Data

- Let’s plot the data
  - Binomial data are tricky to visualize well

- First, organize it as a data frame

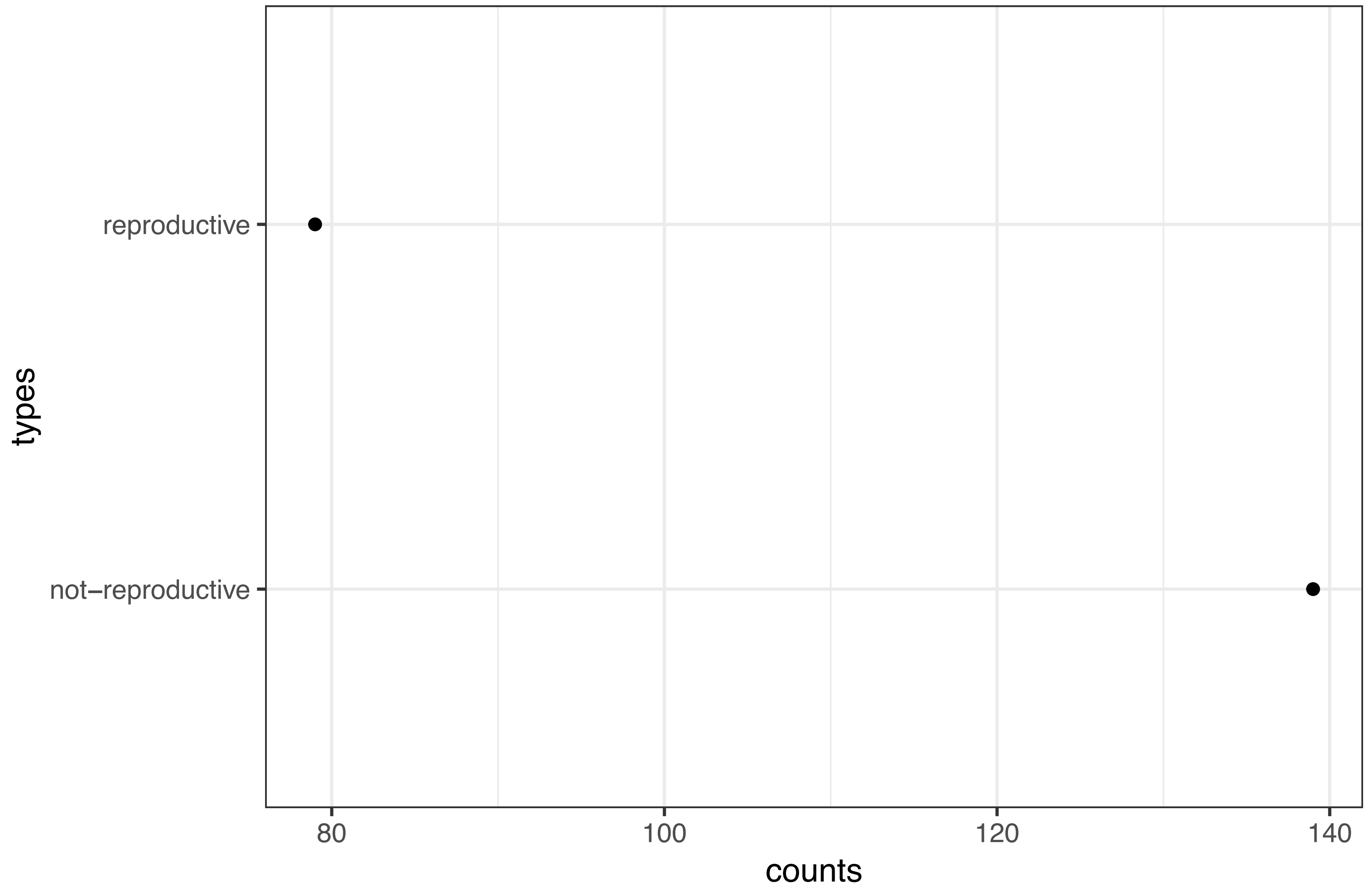
```
types = c("reproductive", "not-reproductive")
counts = c(mothers, females - mothers)
data = data.frame(types, counts)
```

```
data

             types counts
1     reproductive     79
2 not-reproductive    139
```

Data

- Then plot it with ggplot2

```
library(ggplot2)

ggplot(data) +
  theme_bw() +
  geom_point(aes(x = counts, y = types))
```

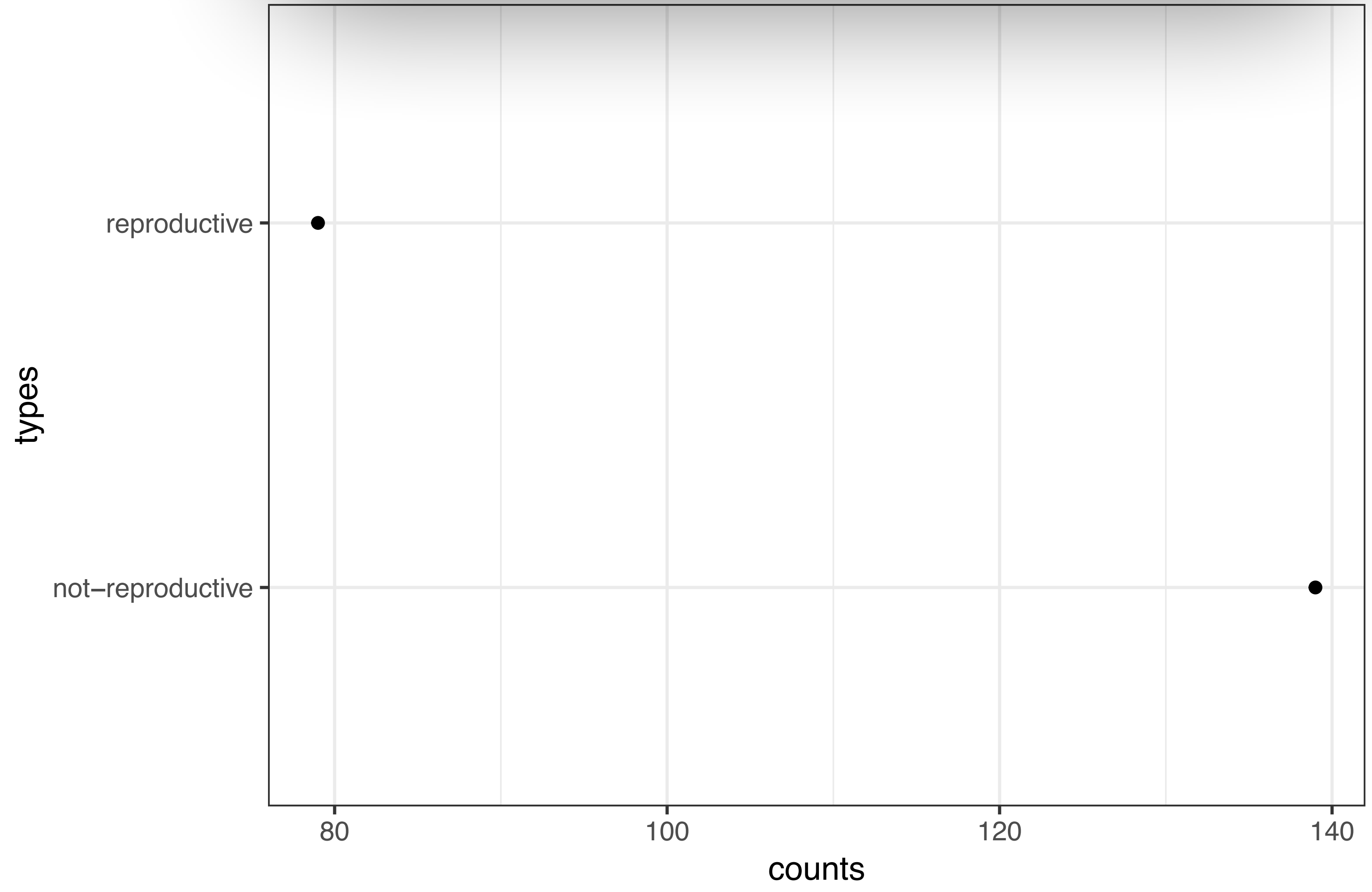


Data

- Then plot it with ggplot2

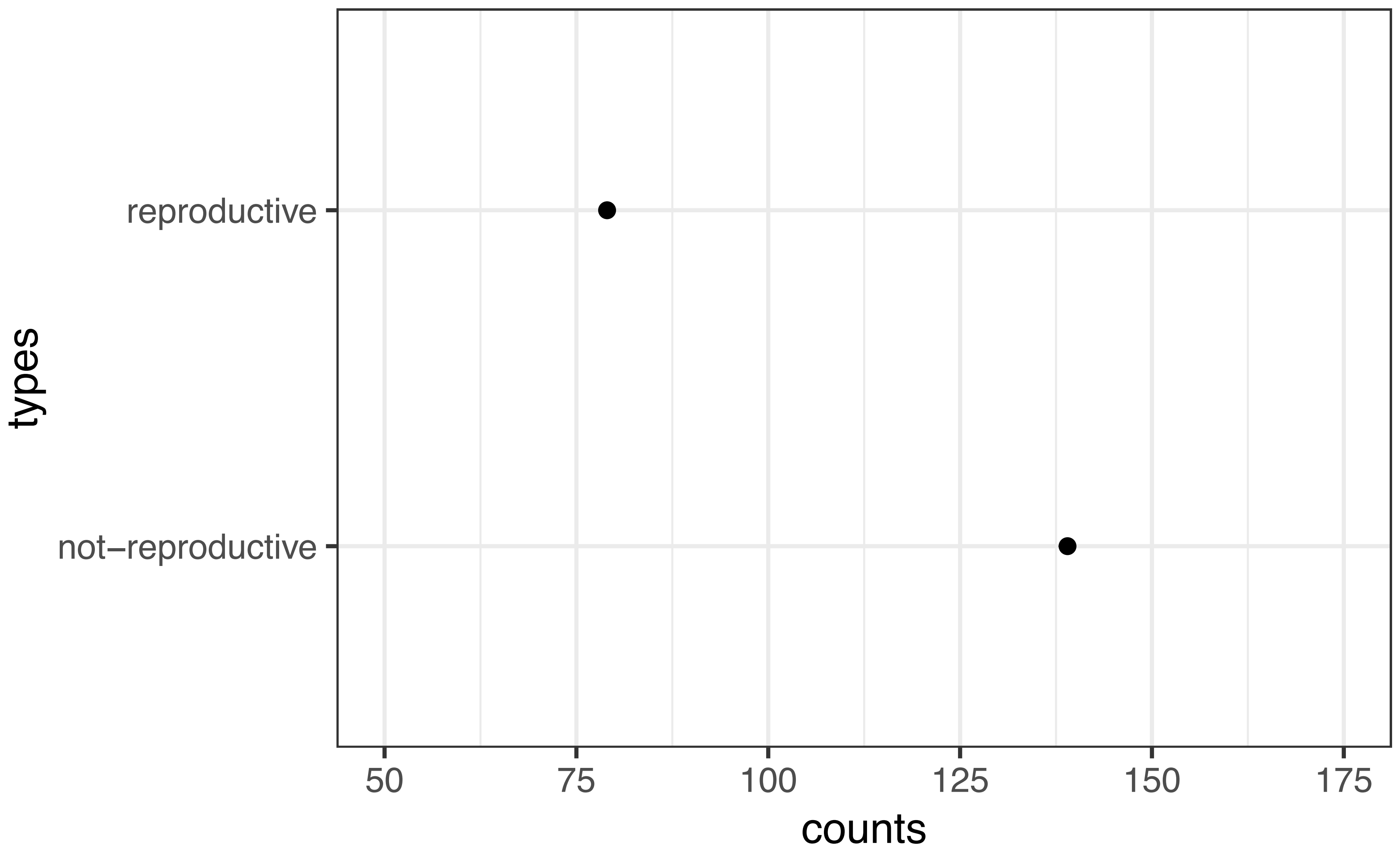
```
library(ggplot2)

ggplot(data) +
  theme_bw() +
  geom_point(aes(x = co
```

Points a little close to the edges. Let's customize the x-axis.



Data

```
ggplot(data) +
  theme_bw() +
  geom_point(aes(x = counts, y = types)) +
  xlim(50, 175)
```



Frequentist Approach

Frequentist Approach

- Such a test is a "binomial test"
- Compare data to some expected value
  - If average inter-birth interval is really 6 years, then in five years ~ 5/6 (= 83%) of adult females should have given birth
- Use the `binom.test` function (three arguments)
  - x = number of successes
  - n = number of trials
  - p (probability of success)

```
binom.test(x, n, p)
```

Frequentist Approach

```
binom.test(x = mothers, n = females, p = 0.83)

	Exact binomial test

data:  mothers and females
number of successes = 79, number of trials = 218, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.83
95 percent confidence interval:
 0.2985498 0.4300200
sample estimates:
probability of success 
             0.3623853 
```

Frequentist Approach

```
binom.test(x = mothers, n = females, p = 0.83)

    Exact binomial test

data:  mothers and females
number of successes = 79, number of trials = 218, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.83
95 percent confidence interval:
 0.2985498 0.4300200
sample estimates:
probability of success
             0.3623853
```

What does this mean again?

Frequentist Approach

```
binom.test(x = mothers, n = females, p = 0.83)

	Exact binomial test

data:  mothers and females
number of successes = 79, number of trials = 218, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.83
95 percent confidence interval:
 0.2985498 0.4300200
sample estimates:
probability of success
             0.3623853
```

95% Confidence Interval

Frequentist Approach

```
binom.test(x = mothers, n = females, p = 0.83)

	Exact binomial test

data:  mothers and females
number of successes = 79, number of trials = 218, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.83
95 percent confidence interval:
 0.2985498 0.4300200
sample estimates:
probability of success
             0.3623853
```

Estimated probability of success

Frequentist Approach

```
binom.test(x = mothers, n = females, p = 0.83)

	Exact binomial test

data:  mothers and females
number of successes = 79, number of trials = 218, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.83
95 percent confidence interval:
 0.2985498 0.4300200
sample estimates:
probability of success
             0.3623853
```

- Hmm…if estimated inter-birth interval (based on the same data) is 6 years, why different?

Bayesian Approach

(Home brew)

Building Our Own MCMC Process

Data

- 79 successes out of 218 trials (139 “failures”)

Question

- What’s the probability of a female reproducing during this time period **and** our **uncertainty** about that probability?

How could we go about addressing this using a simple MCMC approach?

Situation

$$y \sim \text{binomial}(N, \Theta)$$

Situation

$$y \sim \text{binomial}(N, \Theta)$$

Data *- the number of successes* (79)

Situation

$$y \sim \text{binomial}(N, \Theta)$$

Data *- the number of successes* (79)

Data *- the number of "trials"* (218)

Situation

$$y \sim \text{binomial}(N, \Theta)$$

Data *- the number of successes* (79)

Parameter *- the probability of "success"*

Data *- the number of "trials"* (218)

Situation

$$y \sim \text{binomial}(N, \Theta)$$

Parameter *- the probability of "success"*

- All parameters being estimated require a prior
- What seems appropriate?

Situation

$$y \sim \text{binomial}(N, \Theta)$$

$$\Theta \sim \text{uniform}(0, 1)$$

Building Our Own MCMC Process

1. Propose a value for the probability of success
2. Generate a random binomial data set (the same size as ours) using this probability

   ```
   rbinom(n, size, prob)
   rbinom(n = 218, size = 1, prob = ?)
   ```

3. If process results in same number of successes as our data set, record proposed value
   - If not, discard proposed value
4. Repeat steps 1-3 many many times

Building Our Own MCMC Process

1. Propose a value for the probability of success

2. Generate a random binomial data set (the same size as ours) using this probability

```
rbinom(
rbinom(                    )
```

Prior is the ***distribution*** of values that we pull from (not one specific value)

3. If process results in data set, record proposed value
   - If not, discard proposed value

4. Repeat steps 1-3 many many times

Building Our Own MCMC Process

1. Propose a value for the probability of success

2. Generate a random binomial data set (the same size as ours) using this probability

```
rbinom(
rbinom(                                  ?)
```

What seems reasonable?

3. If process results in same number of successes as our data set, record proposed value
    - If not, discard proposed value

4. Repeat steps 1-3 many many times

Building Our Own MCMC Process

1. Propose a value for the probability of success

```
proposal = runif(n = 1, min = 0, max = 1)
proposal
```

Building Our Own MCMC Process

1. Propose a value for the probability of success

```r
proposal = runif(n = 1, min = 0, max = 1)
proposal
```

2. Generate a random binomial data set (the same size as ours) using this probability

```r
simulated = rbinom(n = 218, size = 1, prob = proposal)
```

Building Our Own MCMC Process

1. Propose a value for the probability of success

```
proposal = runif(n = 1, min = 0, max = 1)
proposal
```

2. Generate a random binomial data set (the same size as ours) using this probability

```
simulated = rbinom(n = 218, size = 1, prob = proposal)
```

3. See if it results in the same number of success as our data

```
sum(simulated)
```

Building Our Own MCMC Process

1. Propose a value for the probability of success

```
proposal = runif(n = 1, min = 0, max = 1)
proposal
```

2. Generate a random binomial data set (the same size as ours) using this probability

```
simulated = rbinom(n = 218, size = 1, prob = proposal)
```

3. See if it results in the same number of success as our data

```
sum(simulated)
```

4. Repeat again and again

Building Our Own MCMC Process

- Load code for automating this

```
source("binomialSim.R")
```

- Function is called `binomialCalc`, requires 3 arguments

```
binomialCalc(nTrials, nSuccesses, nSteps)
```

Demo!

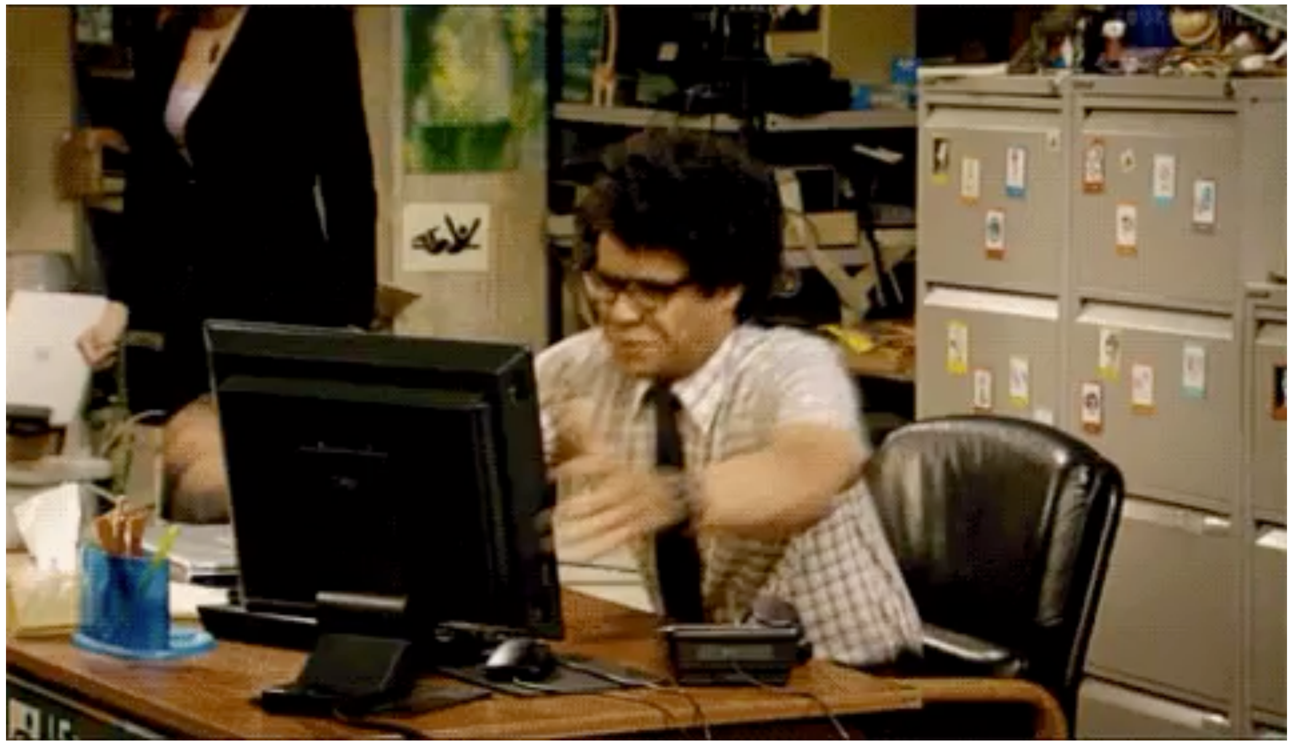
Building Our Own MCMC Process

- This method is super inefficient, no?
- Why “smart” MCMC samplers are important

Bayesian Approach

(Stan)

Sometimes how you feel when programming R and Stan



Using Raw Stan and R

- Are many packages to make using Stan, and visualizing results, easier
- We won’t use them, but will get our hands dirty “under the hood” with the real code

Analyses With Stan (or any MCMC process)

1. Prepare data for Stan
2. Build/define model
3. Run model
4. Assess MCMC process
5. Tentatively evaluate results
6. Conduct posterior predictive checks
7. Accept results or go back to step 2 to refine model

1. Prepare data for Stan

Remember

$$y \sim \text{binomial}(N, \Theta)$$

$$\Theta \sim \text{uniform}(0, 1)$$

1. Prepare Data for Stan

- Have to pass data to Stan as a `list`

```
dataList = list (
  y = mothers,
  N = females
)
```

*Must include **all** of the data Stan will need*

$$y \sim \text{binomial}(N, \Theta)$$

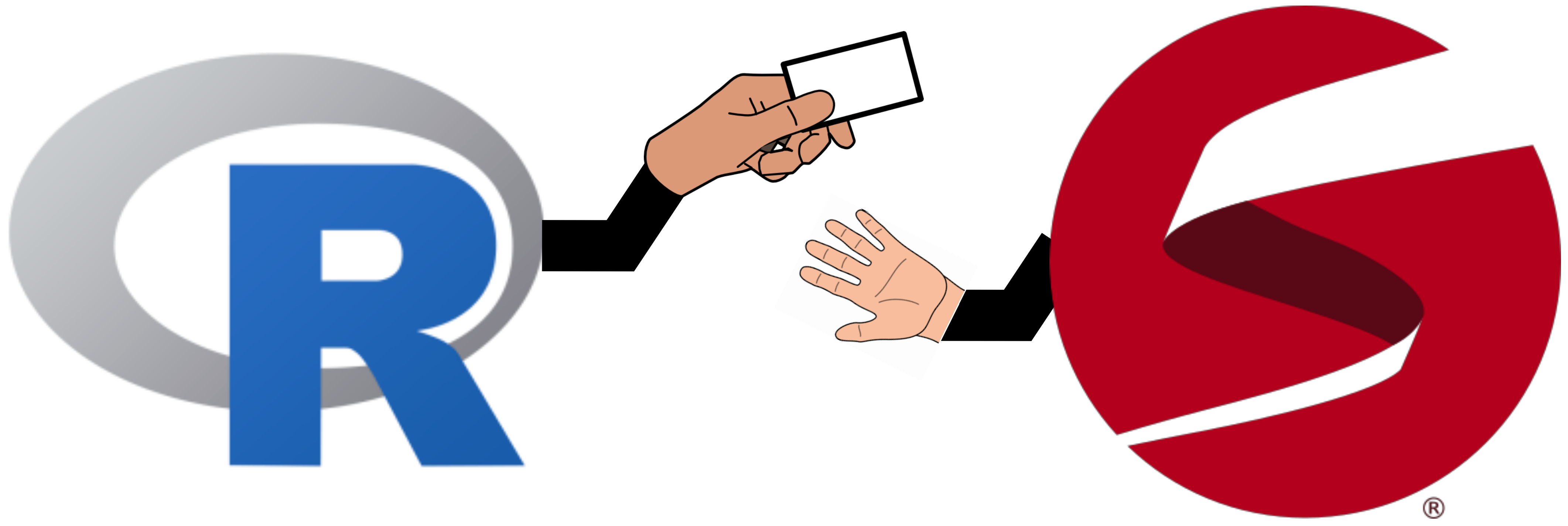
$$\Theta \sim \text{uniform}(0, 1)$$

1. Prepare Data for Stan

- Have to pass data to Stan as a `list`

```
dataList = list (
  y = mothers,
  N = females
)
```

*Must include **all** of the data Stan will need*



2. Build/Define the Model

2. Build/Define The Model

- Stan written in C++
    - Need to write our model in C++
        1. Lines where things are declared/assigned must end in a semicolon (`;`)
        2. Ignores lines starting with `//` (used for comments)

2. Build/Define The Model

- Needs a minimum of three “blocks” (**in order!!**)
    1. **Data block**: define **all** of the data your model needs to run
    2. **Parameters block**: define **all** of the parameters you will be estimating in your model
    3. **Model block**: define likelihood and priors
- However, I think it is easier to build them in reverse order

2. Build/Define The Model

- Define the “model” block
  - Define the likelihood(s) and prior(s)

```
model {
  // Likelihood
  y ~ binomial(N, theta);

  // Priors
  theta ~ uniform(0, 1);
}
```

$$y \sim \text{binomial}(N, \Theta)$$

$$\Theta \sim \text{uniform}(0, 1)$$

2. Build/Define The Model

- Define the "model" block
    - Define the likelihood(s) and prior(s)

```
model {
  // Likelihood
  y ~ binomial(N, theta);

  // Priors
  theta ~ uniform(0, 1);
}
```

"Calculate the likelihood of obtaining our `y` data (79) from a binomial distribution of size `N`, and with a probability of "success" of `theta`."

2. Build/Define The Model

- Define the “model” block
  - Define the likelihood(s) and prior(s)

```
model {
  // Likelihood
  y ~ binomial(N, theta);

  // Priors
  theta ~ uniform(0, 1);
}
```

“Propose `theta` values by pulling them randomly from a uniform distribution from 0 to 1.”

2. Build/Define The Model

- Define the “model” block
  - Define the likelihood(s) and prior(s)

```
model {
   // Likelihood
   y ~ binomial(N, theta);

   // Priors
   theta ~ uniform(0, 1);
}
```

Anything being estimated from, or being drawn from, a distribution requires a ~ rather than = !!!!! It samples “sampled from”

2. Build/Define The Model

- Define the “parameters” block
  - Define **all** of the parameters you will be estimating in your model

```
parameters {
  real theta;
}
```

$$y \sim \text{binomial}(N, \Theta)$$

$$\Theta \sim \text{uniform}(0, 1)$$

2. Build/Define The Model

- Define the “parameters” block
    - Define **all** of the parameters you will be estimating in your model

```
parameters {
  real theta;
}
```

Is a “real” number (can have decimal places).

2. Build/Define The Model

- Define the "data" block
  - Define **all** of the data your model needs to run

```
data {
  int<lower=0> N;
  int<lower=0> y;
}
```

$$y \sim \text{binomial}(N, \Theta)$$

$$\Theta \sim \text{uniform}(0, 1)$$

2. Build/Define The Model

- Define the "data" block
  - Define **all** of the data your model needs to run

```
data {
  int<lower=0> N;
  int<lower=0> y;
}
```

Other possibilities:

- `real`
- `vector`
- `matrix`
- ...

2. Build/Define The Model

- Define the "data" block
  - Define **all** of the data your model needs to run

```
data {
  int<lower=0> N;
  int<lower=0> y;
}
```

Can set lower and upper boundaries

2. Build/Define The Model

- Define the "data" block
  - Define **all** of the data your model needs to run

```
data {
   int<lower=0> N;
   int<lower=0> y;
}
```

Can only refer to parameters you sent to Stan in your previous list!!!

2. Build/Define The Model

- Have to save as a string
  - Declare as a variable, and wrapped in quotes
  - Write as a `.stan` file

```
modelString = "
  data {
    int<lower=0> N;
    int<lower=0> y;
  }

  parameters {
    real theta;
  }

  model {
    // Likelihood
    y ~ binomial(N, theta);

    // Priors
    theta ~ uniform(0, 1);
  }
"
writeLines(modelString, con="model.stan")
```

3. Run the Model

3. Run The Model

```
stanFit = stan(file = "model.stan",
                data = dataList,
                pars = "theta",
                warmup = 2000,
                iter = 5000,
                chains = 3)
```

3. Run The Model

```
stanFit = stan(file = "model.stan",
               data = dataList,
               pars = "theta",
               warmup = 2000,
               iter = 5000,
               chains = 3)
```

The name of the `.stan` file that we just created

3. Run The Model

```
stanFit = stan(file = "model.stan",
               data = dataList,
               pars = "theta",
               warmup = 2000,
               iter = 5000,
               chains = 3)
```

The name of the list containing our data.

3. Run The Model

```
stanFit = stan(file = "model.stan",
               data = dataList,
               pars = "theta",
               warmup = 2000,
               iter = 5000,
               chains = 3)
```

The parameters that we want to save results for.

3. Run The Model

```
stanFit = stan(file = "model.stan",
               data = dataList,
               pars = "theta",
               warmup = 2000,
               iter = 5000,
               chains = 3)
```

How many steps to allow for the "warmup"

3. Run The Model

```
stanFit = stan(file = "model.stan",
               data = dataList,
               pars = "theta",
               warmup = 2000,
               iter = 5000,
               chains = 3)
```

How many iterations (steps) to run the model for (includes the warmup).

3. Run The Model

```
stanFit = stan(file = "model.stan",
               data = dataList,
               pars = "theta",
               warmup = 2000,
               iter = 5000,
               chains = 3)
```

How many chains to run.

3. Run The Model

```
stanFit = stan(file = "model.stan",
               data = dataList,
               pars = "theta",
               warmup = 2000,
               iter = 5000,
               chains = 3)
```

- Will compile this code (takes a while)
  - No response to screen
- Actual running will be fast, and will write to screen

4. Assess Performance of MCMC Process

4. Assess MCMC Process

- Was warmup long-enough?
    - Did chains centre around a value prior to warmup ending?
- Were chains long enough?
    - Did chains centre around a single value long before chains ended?
- Did all chains find the same peak?
    - Do chains centre around the same value

4. Assess MCMC Process

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5%
theta    0.36    0.00 0.03    0.30    0.34    0.36    0.39    0.43
lp__  -143.26    0.01 0.74 -145.38 -143.45 -142.97 -142.79 -142.74
      n_eff Rhat
theta  2652    1
lp__   3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

4. Assess MCMC Process

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5%
theta    0.36    0.00 0.03    0.30    0.34    0.36    0.39    0.43
lp__  -143.26    0.01 0.74 -145.38 -143.45 -142.97 -142.79 -142.74
      n_eff Rhat
theta  2652    1
lp__   3230    1

Samples were dra                                     n 29 14:33:32 2019.
For each paramet                                      effective sample size,
and Rhat is the p                                    or on split chains (at
convergence, Rhat=1).
```

# of chains x post warmup length of chains

4. Assess MCMC Process

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5%
theta    0.36    0.00 0.03    0.30    0.34    0.36    0.39    0.43
lp__  -143.26    0.01 0.74 -145.38 -143.45 -142.97 -142.79 -142.74
      n_eff Rhat
theta  2652    1
lp__   3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential sc                             hains (at
convergence, Rhat=1).
```

Summary statistics for the parameter(s) of interest

4. Assess MCMC Process

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

          mean se_mean   sd    2.5%     25%     50%     75%   97.5%
theta     0.36    0.00 0.03    0.30    0.34    0.36    0.39    0.43
lp__   -143.26    0.01 0.74 -145.38 -143.45 -142.97 -142.79 -142.74
       n_eff Rhat
theta   2652    1
lp__    3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential sc                                hains (at
convergence, Rhat=1).
```

Are called the **Highest Density Intervals** (HDI). Can be interpreted they way you think confidence intervals should.

4. Assess MCMC Process

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5%
theta    0.36    0.00 0.03    0.30    0.34    0.36    0.39    0.43
lp__  -143.26    0.01 0.74 -145.38 -143.45 -142.97 -142.79 -142.74
      n_eff Rhat
theta  2652    1
lp__   3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential sc                               hains (at
convergence, Rhat=1).
```

Same data for the log-posterior

4. Assess MCMC Process

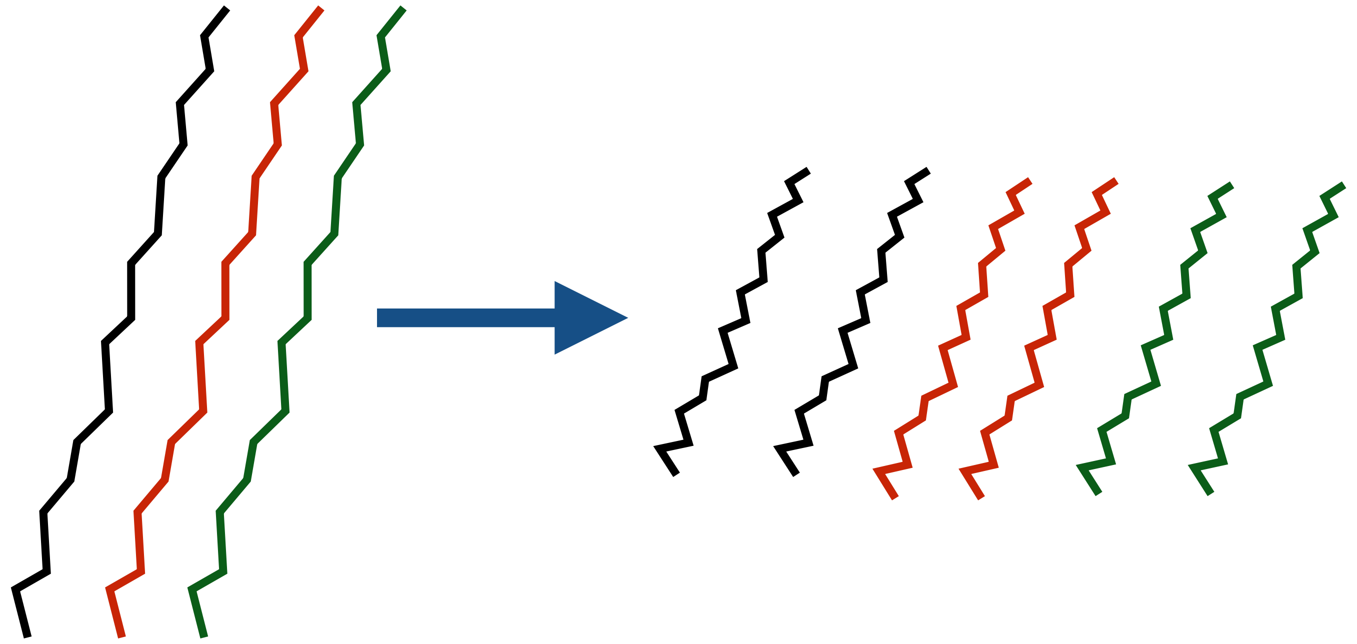
```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5%
theta    0.36    0.00 0.03    0.30    0.34    0.36    0.39    0.43
lp__  -143.26    0.01 0.74 -145.38 -143.45 -142.97 -142.79 -142.74
      n_eff Rhat
theta  2652    1
lp__   3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Main diagnostic we are looking for

4. Assess MCMC Process

Rhat

- Split each chain in half



- Take ratio of between chain variance:within chain variance
  - If chains have explored the same space, should be 1
  - If chains have explored different space (are on different peaks), will be >1
  - If Rhat < 1.1, you're probably OK

4. Assess MCMC Process

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5%
theta    0.36    0.00 0.03    0.30    0.34    0.36    0.39    0.43
lp__  -143.26    0.01 0.74 -145.38 -143.45 -142.97 -142.79 -142.74
      n_eff Rhat
theta  2652    1
lp__   3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

So far, so good!

4. Assess MCMC Process

```
print(stanFit)

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5%
theta    0.36    0.00 0.03    0.30    0.34    0.36    0.39    0.43
lp__  -143.26    0.01 0.74 -145.38 -143.45 -142.97 -142.79 -142.74
      n_eff Rhat
theta  2652    1
lp__   3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter,
and Rhat is the pot
convergence, Rhat=1
```

Effective chain length

MCMC chains always have some autocorrelation (non-independence between steps). Estimates the number of "independent" steps given the autocorrelation.

4. Assess MCMC Process

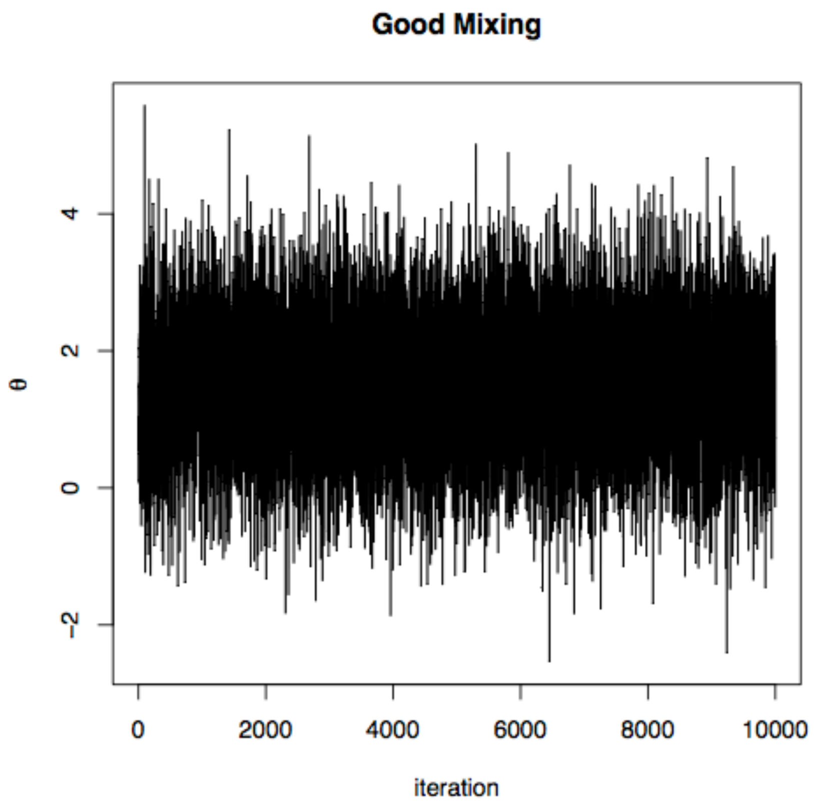
- Can customize summary statistics

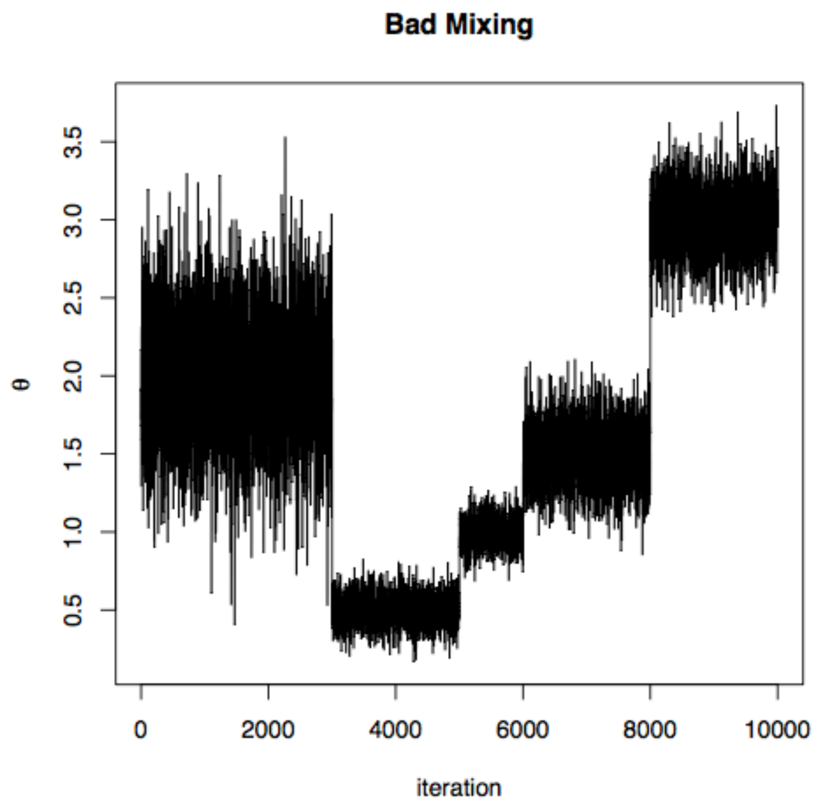
```
print(stanFit, probs = c(0.055, 0.5, 0.945))

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    5.5%     50%   94.5% n_eff Rhat
theta    0.36    0.00 0.03    0.31    0.36    0.42  2652    1
lp__  -143.26    0.01 0.74 -144.70 -142.97 -142.74  3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

4. Assess MCMC Process

Traceplots

- Plots iteration # against value drawn for that iteration
- If chain is mixing well, should look like a wide mess that is consistent "noise" around a mean value - a "spiky caterpillar"

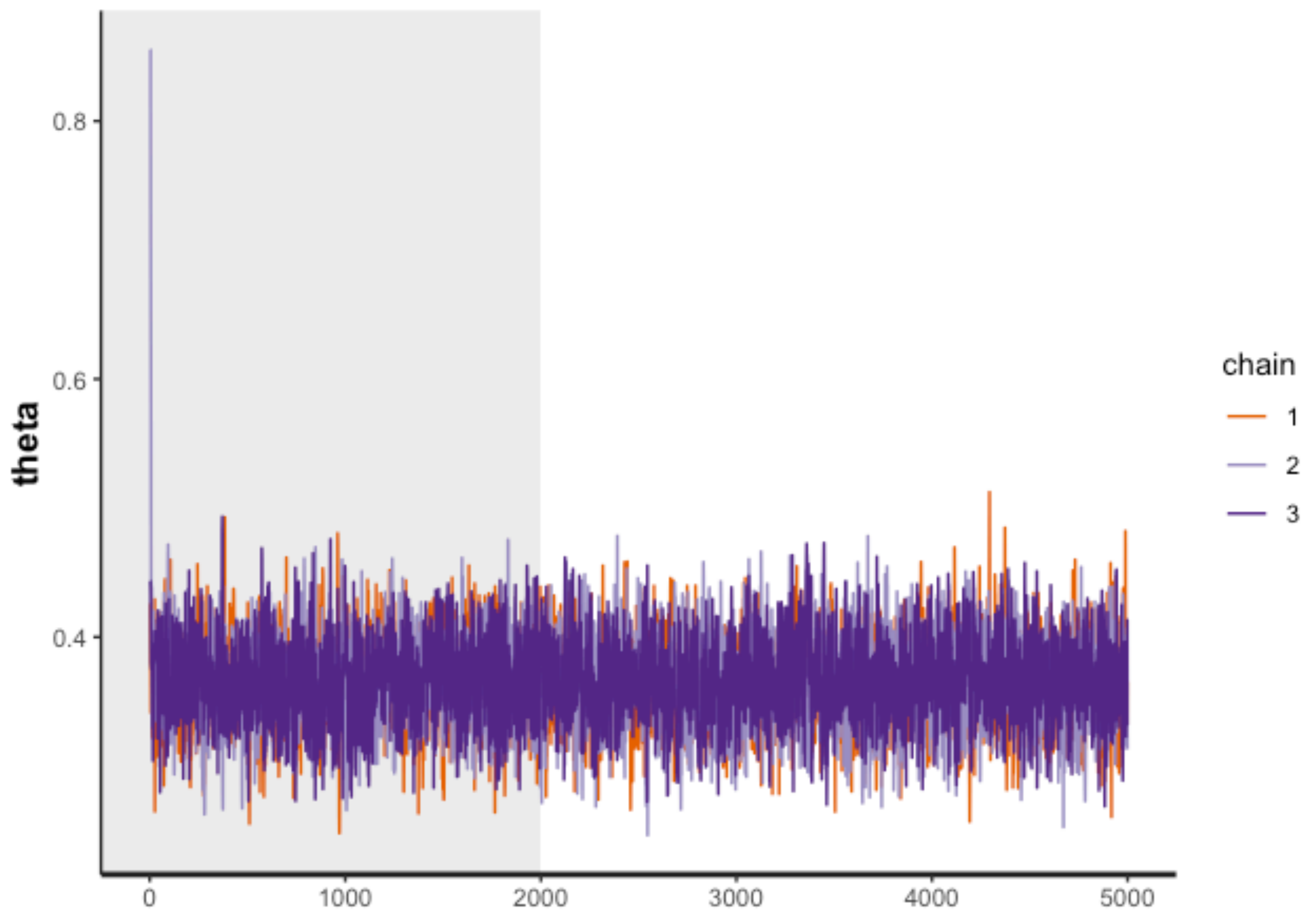




4. Assess MCMC Process

Traceplots

```
stan_trace(stanFit, pars = "theta", inc_warmup = TRUE)
```

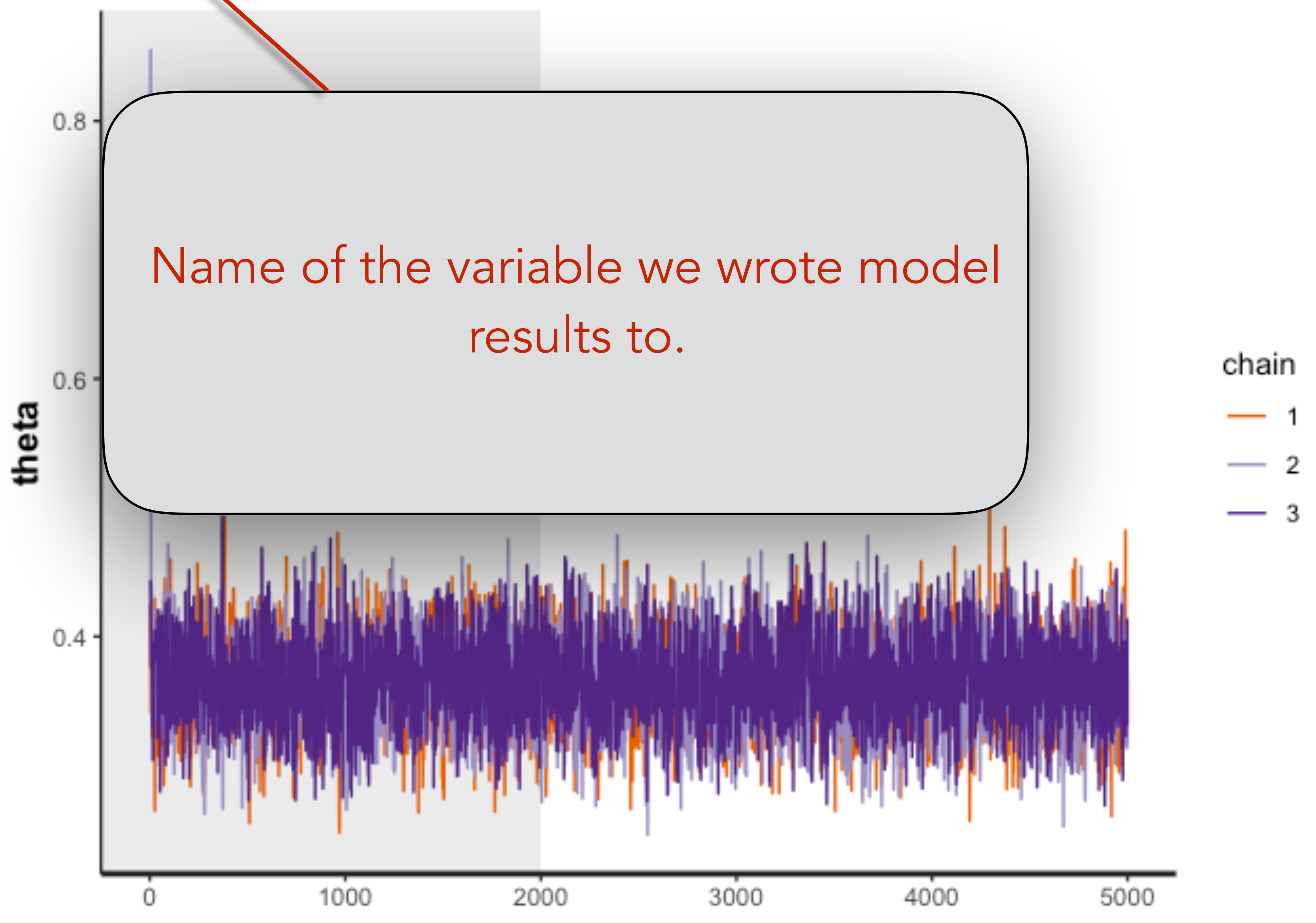


4. Assess MCMC Process

Traceplots

```
stan_trace(stanFit, pars = "theta", inc_warmup = TRUE)
```

Name of the variable we wrote model results to.



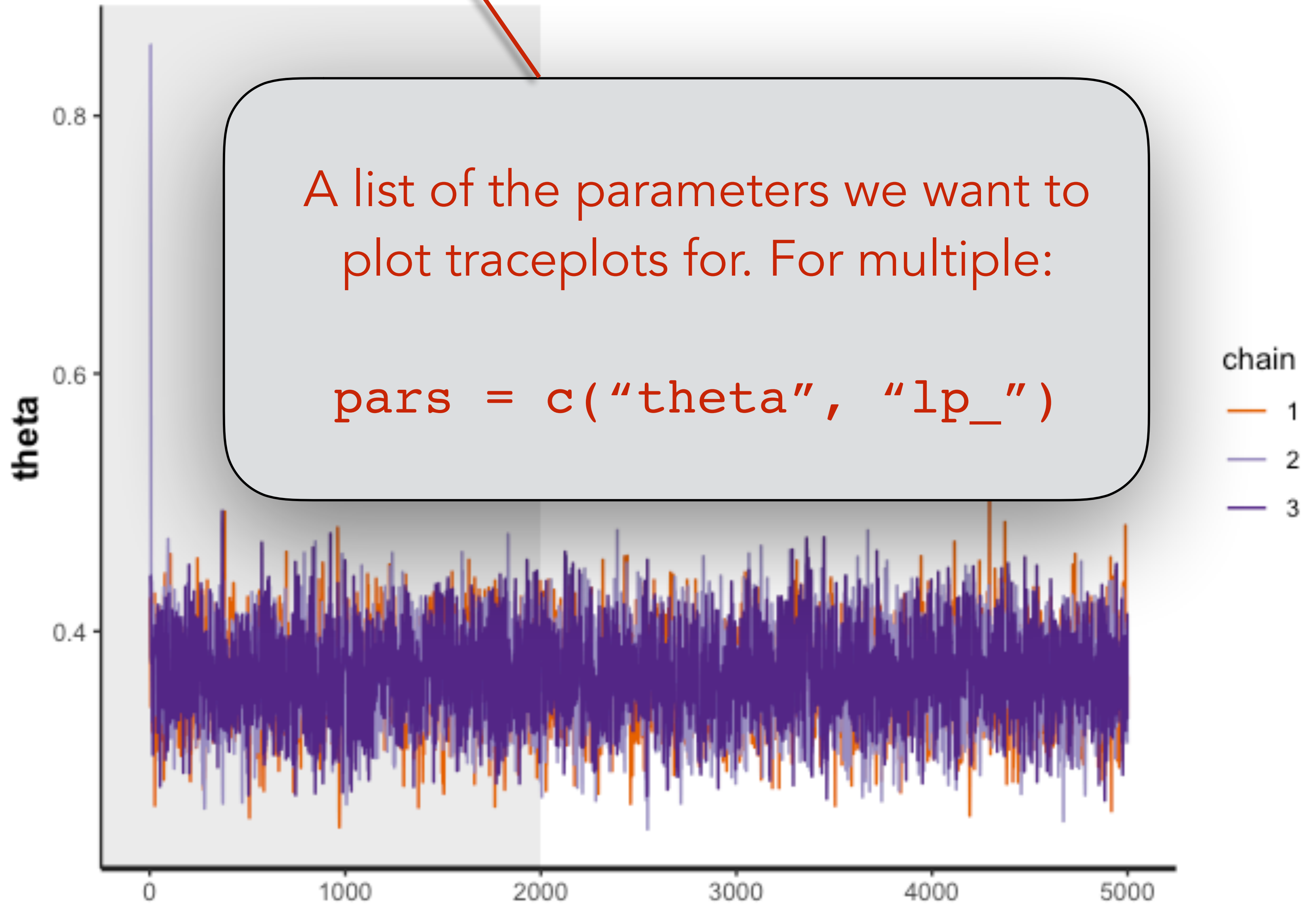
4. Assess MCMC Process

Traceplots

```
stan_trace(stanFit, pars = "theta", inc_warmup = TRUE)
```

A list of the parameters we want to plot traceplots for. For multiple:

```
pars = c("theta", "lp_")
```

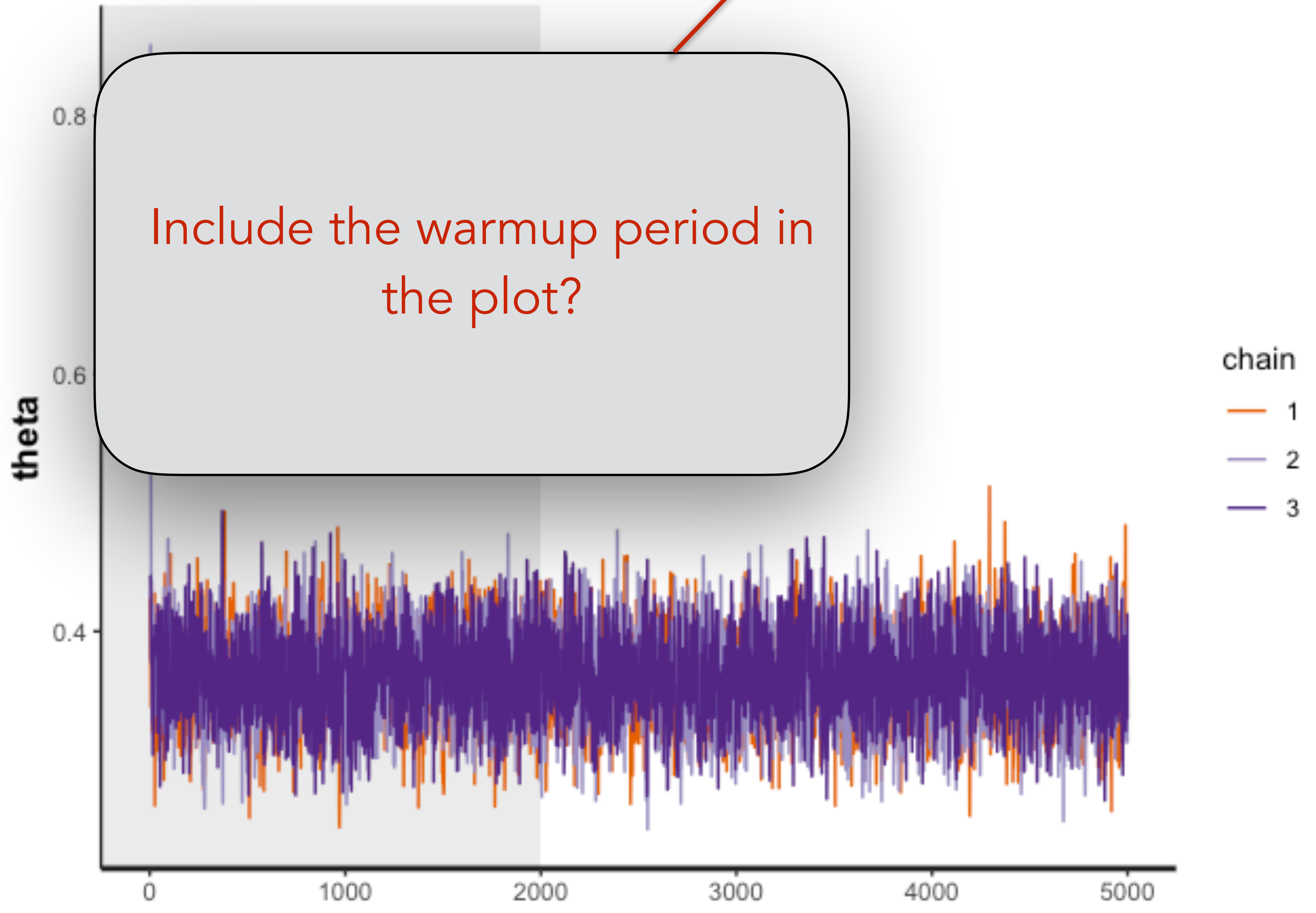


4. Assess MCMC Process

Traceplots

```
stan_trace(stanFit, pars = "theta", inc_warmup = TRUE)
```

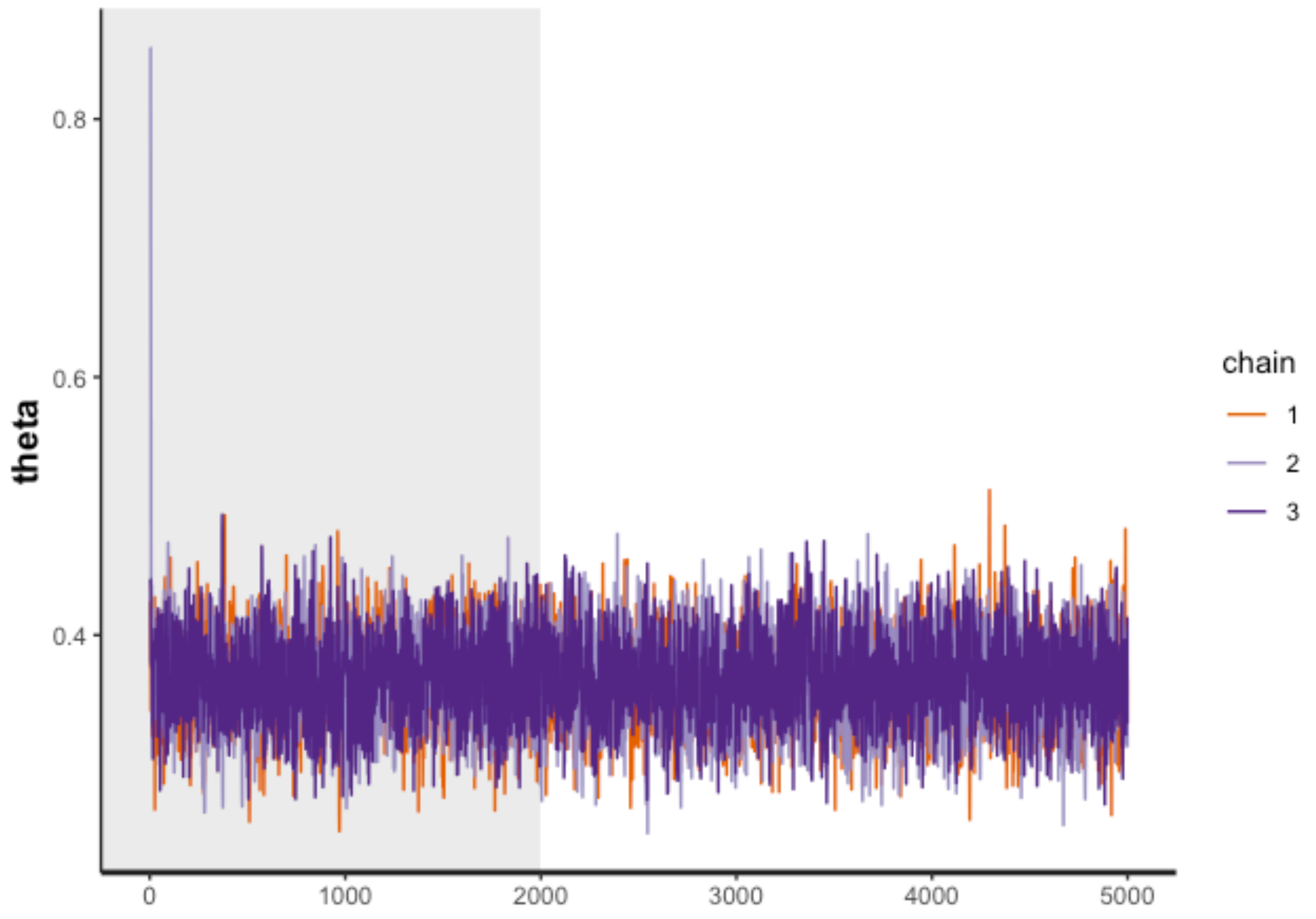
Include the warmup period in the plot?



4. Assess MCMC Process

Traceplots

- Looks good!



4. Assess MCMC Process

- Was warmup long-enough?
  - Did chains centre around a value prior to warmup ending? ✓
- Were chains long enough?
  - Did chains centre around a single value long before chains en ✓
- Did all chains find the same peak?
  - Do chains centre around the same value ✓

5. Tentatively Interpret Results

5. Tentatively Interpret Results

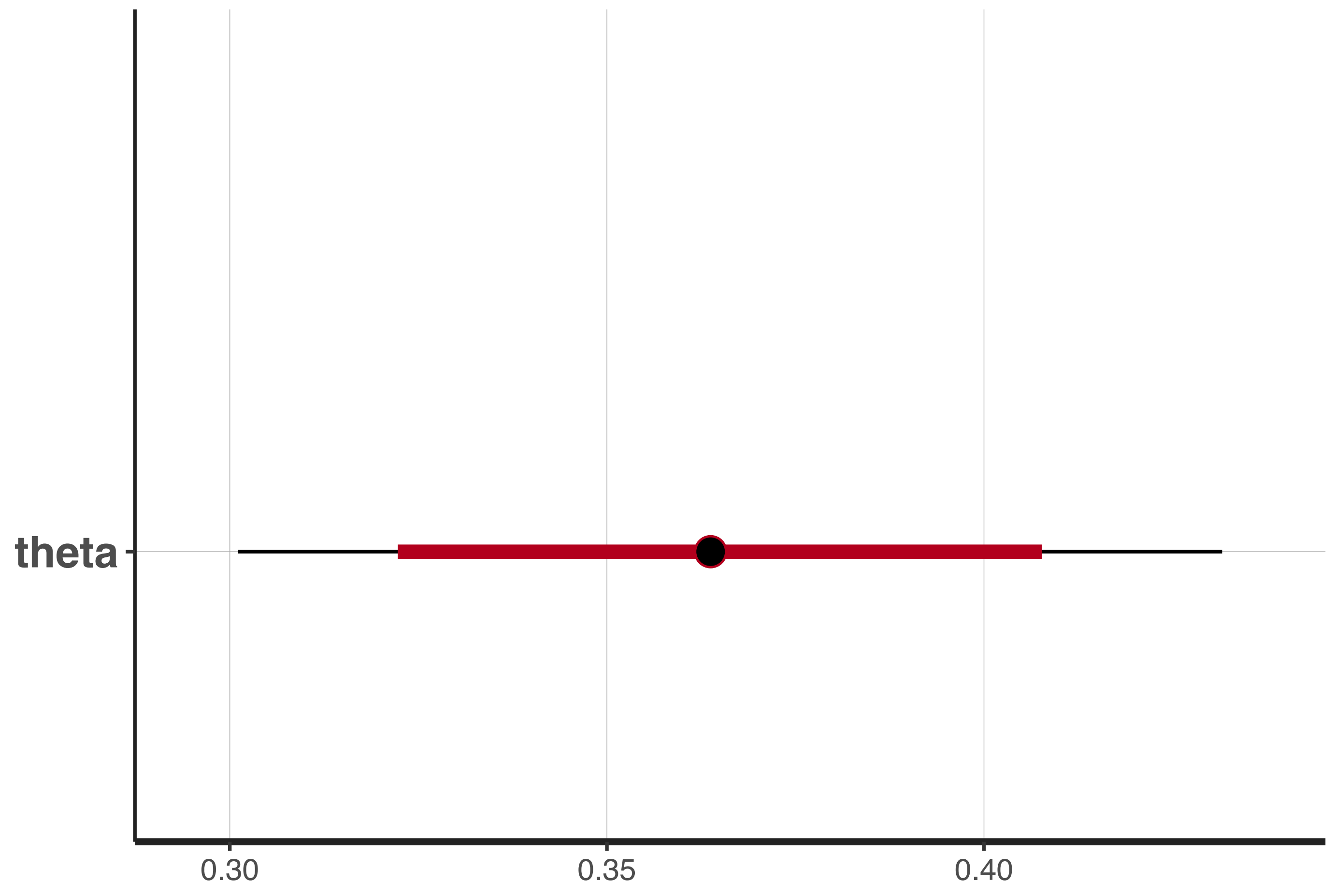
View stats

```
print(stanFit, probs = c(0.055, 0.5, 0.945))

Inference for Stan model: model.
3 chains, each with iter=5000; warmup=2000; thin=1;
post-warmup draws per chain=3000, total post-warmup draws=9000.

         mean se_mean   sd    5.5%     50%   94.5% n_eff Rhat
theta    0.36    0.00 0.03    0.31    0.36    0.42  2652    1
lp__  -143.26    0.01 0.74 -144.70 -142.97 -142.74  3230    1

Samples were drawn using NUTS(diag_e) at Tue Jan 29 14:33:32 2019.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

5. Tentatively Interpret Results

Plot with `rstan` functions

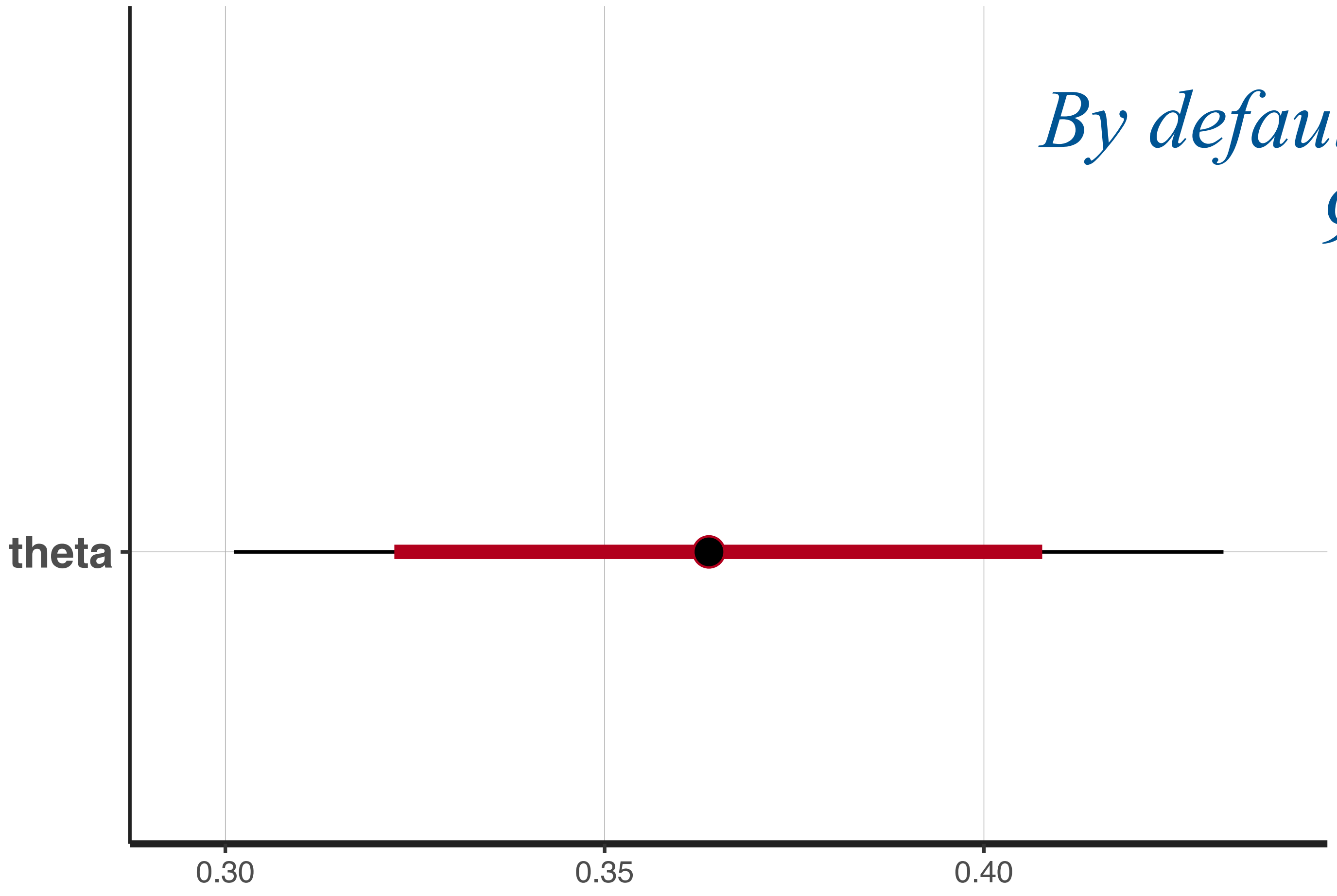
```
stan_plot(stanFit, par = "theta")
```



5. Tentatively Interpret Results

Plot with `rstan` functions

```
stan_plot(stanFit, par = "theta")
```



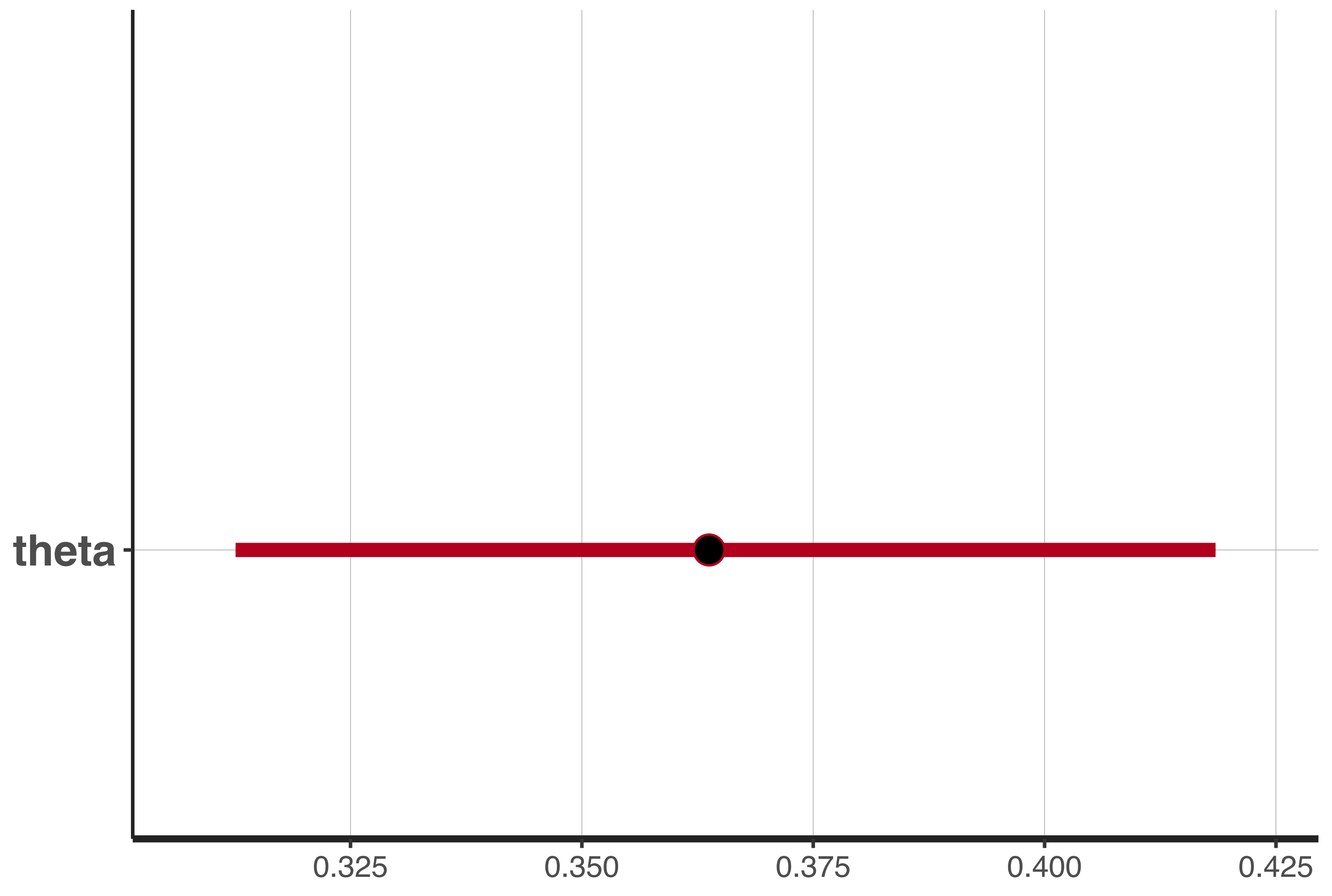
By default shows 80% and 95% HDI

5. Tentatively Interpret Results

Plot with `rstan` functions

- Can customize (see `?stan_plot`)

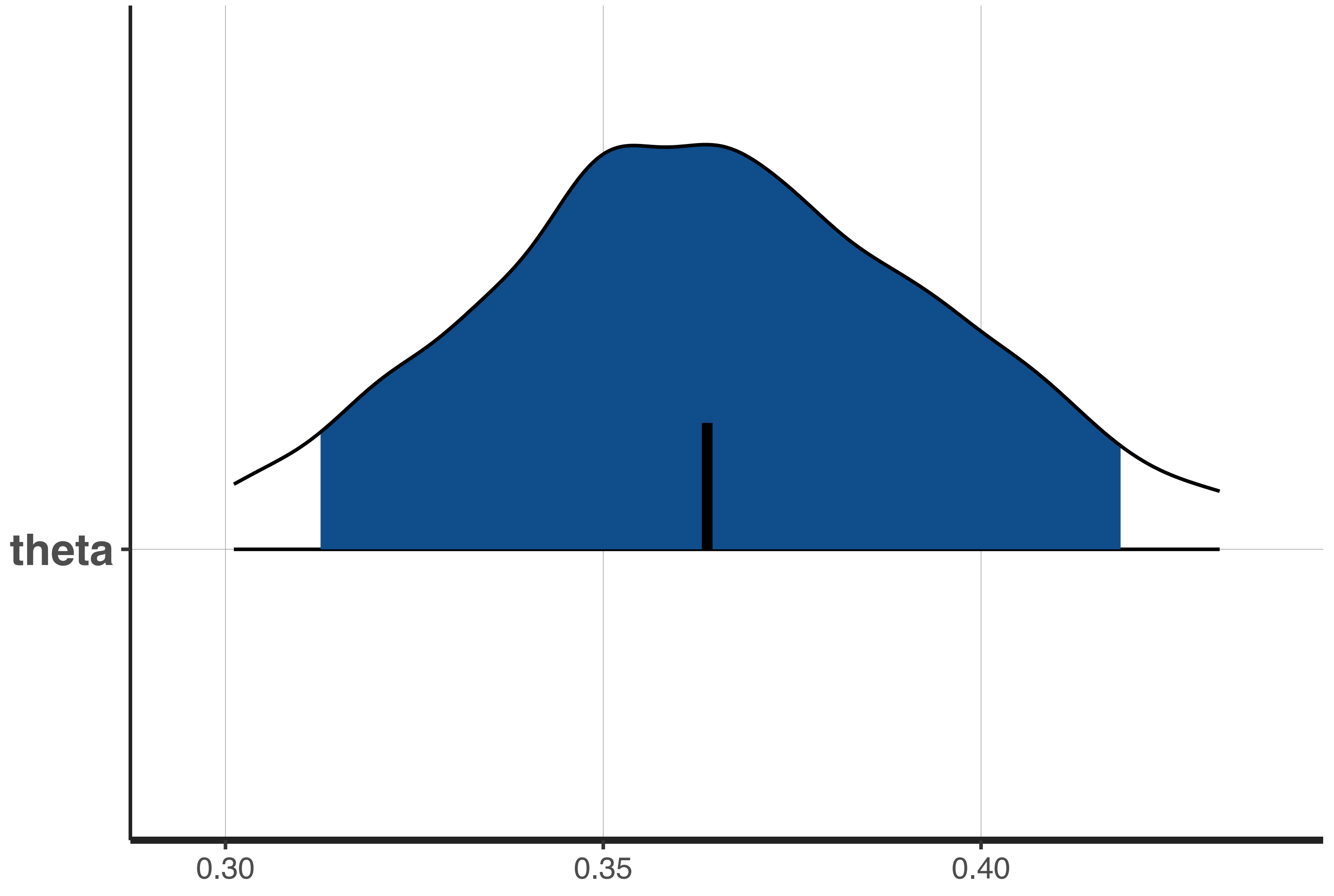
```
stan_plot(stanFit, par = "theta", ci_level = 0.89, outer_level = 0.89)
```



5. Tentatively Interpret Results

Plot with `rstan` functions

```
plot(stanFit, par = "theta", show_density = TRUE, ci_level = 0.89,
     fill_color = "dodgerblue4")
```



5. Tentatively Interpret Results

Make custom plots with ggplot2

- Extract the data as a data frame and take a look at it’s structure

```
posteriors = as.data.frame(stanFit)
str(posteriors)

'data.frame':  9000 obs. of  2 variables:
 $ theta: num  0.352 0.376 0.379 0.358 0.341 ...
 $ lp__ : num  -143 -143 -143 -143 -143 ...
```

5. Tentatively Interpret Results

Make custom plots with ggplot2

- Take a peak to ensure it looks how you expect

```
head(posteriors)

      theta       lp__
1 0.3520303 -142.7927
2 0.3760412 -142.8287
3 0.3788629 -142.8682
4 0.3580252 -142.7507
5 0.3413926 -142.9533
6 0.3240659 -143.4586
```

5. Tentatively Interpret Results
Make custom plots with ggplot2

- Calculate desired summary statistics (for plotting)

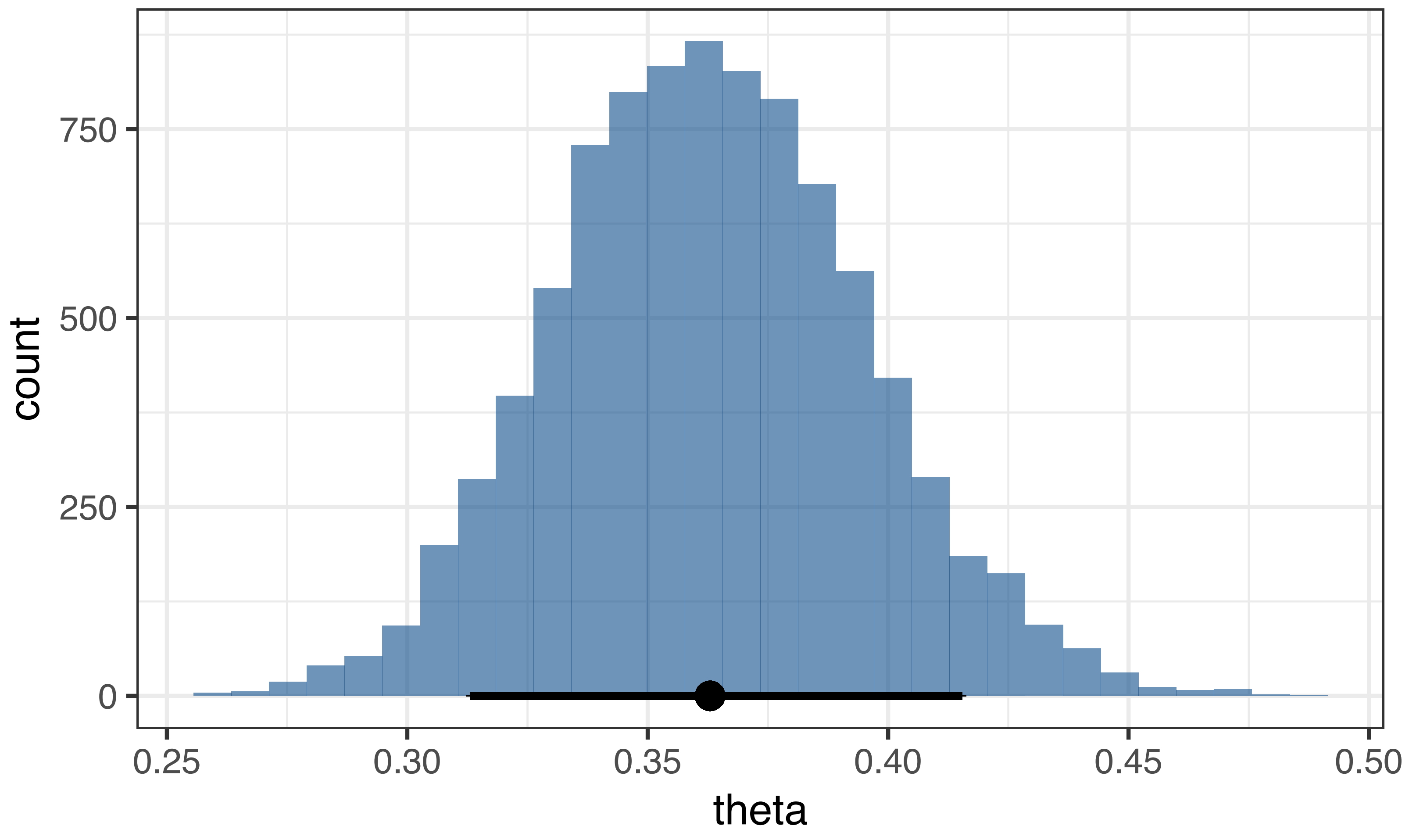
```
postMean = mean(posteriors$theta)
limits = quantile(posteriors$theta, probs = c(0.055, 0.945))
limits

     5.5%     94.5%
0.3130937 0.4153827
```

```
postHDI = as.numeric(limits)
postHDI

[1] 0.3127329 0.4139767
```

5. Tentatively Interpret Results

Plot as a histogram

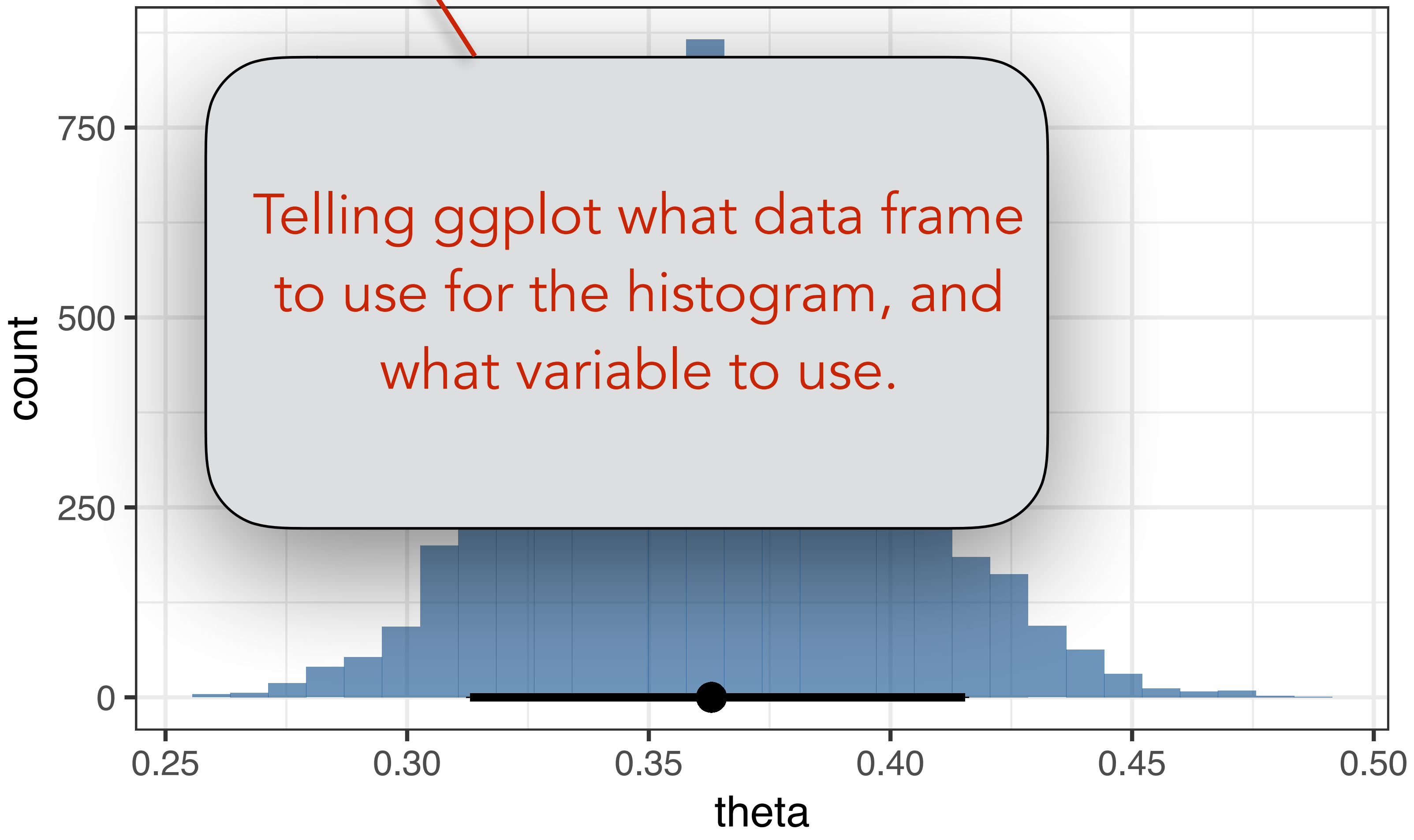
```
ggplot(posteriors, aes(x = theta)) +
  theme_bw() +
  geom_histogram(fill = "dodgerblue4", alpha = 0.6) +
  geom_point(aes(x = postMean, y = 0), size = 3) +
  geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
    size = 1)
```



5. Tentatively Interpret Results

Plot as a histogram

```
ggplot(posteriors, aes(x = theta)) +
    theme_bw() +
    geom_histogram(fill = "dodgerblue4", alpha = 0.6) +
    geom_point(aes(x = postMean, y = 0), size = 3) +
    geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
      size = 1)
```

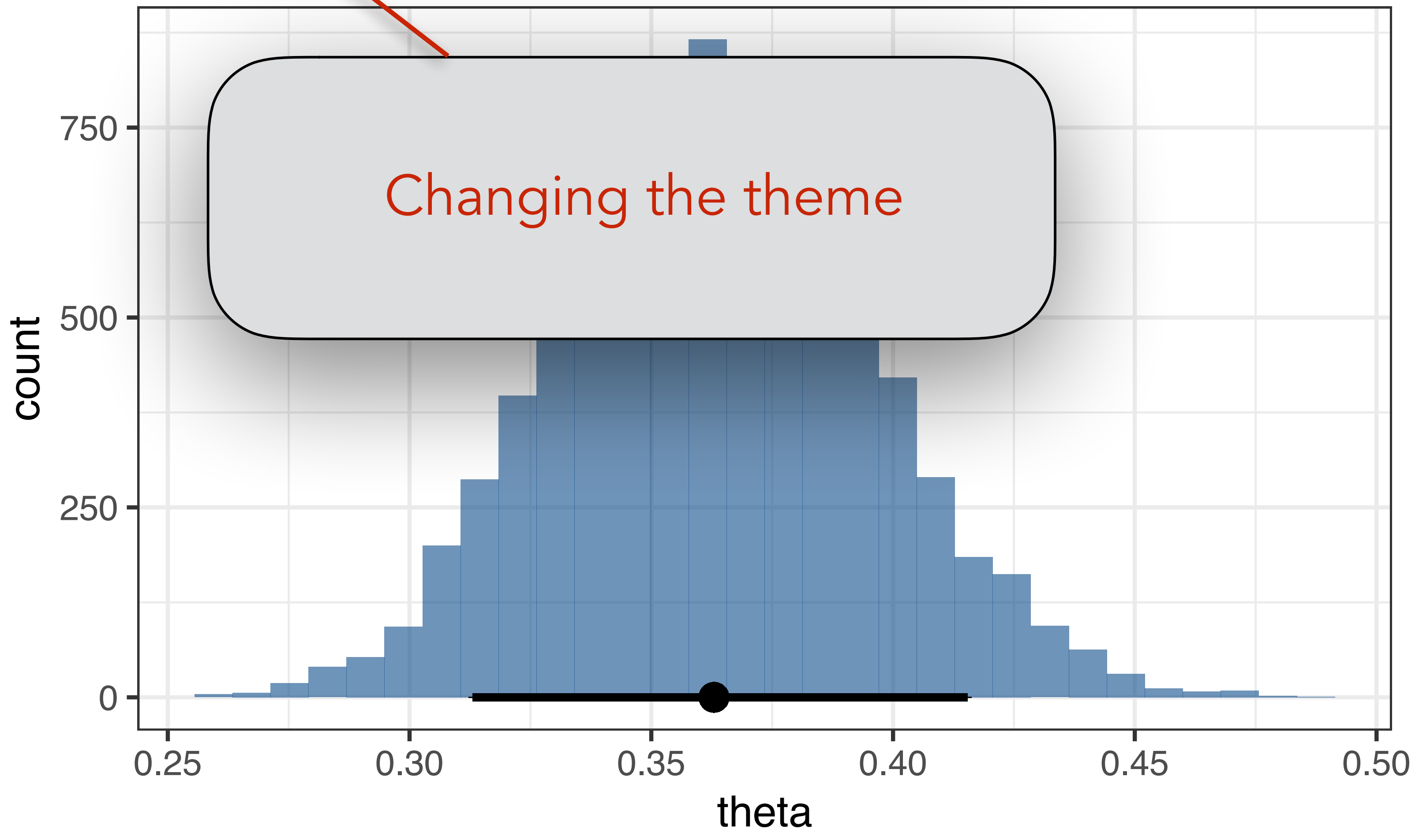


Telling ggplot what data frame to use for the histogram, and what variable to use.

5. Tentatively Interpret Results

Plot as a histogram

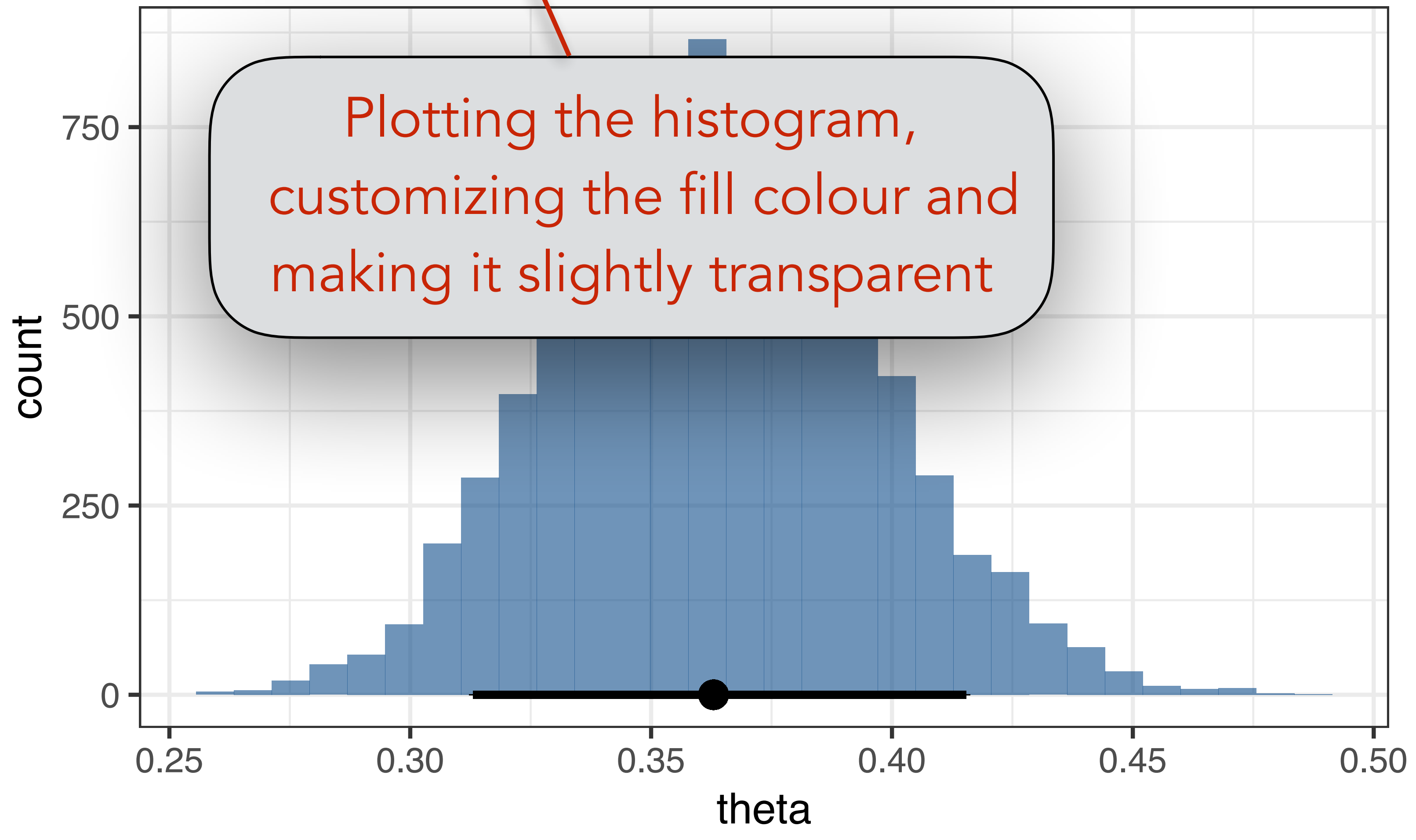
```
ggplot(posteriors, aes(x = theta)) +
  theme_bw() +
  geom_histogram(fill = "dodgerblue4", alpha = 0.6) +
  geom_point(aes(x = postMean, y = 0), size = 3) +
  geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
    size = 1)
```



5. Tentatively Interpret Results

Plot as a histogram

```
ggplot(posteriors, aes(x = theta)) +
  theme_bw() +
  geom_histogram(fill = "dodgerblue4", alpha = 0.6) +
  geom_point(aes(x = postMean, y = 0), size = 3) +
  geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
    size = 1)
```

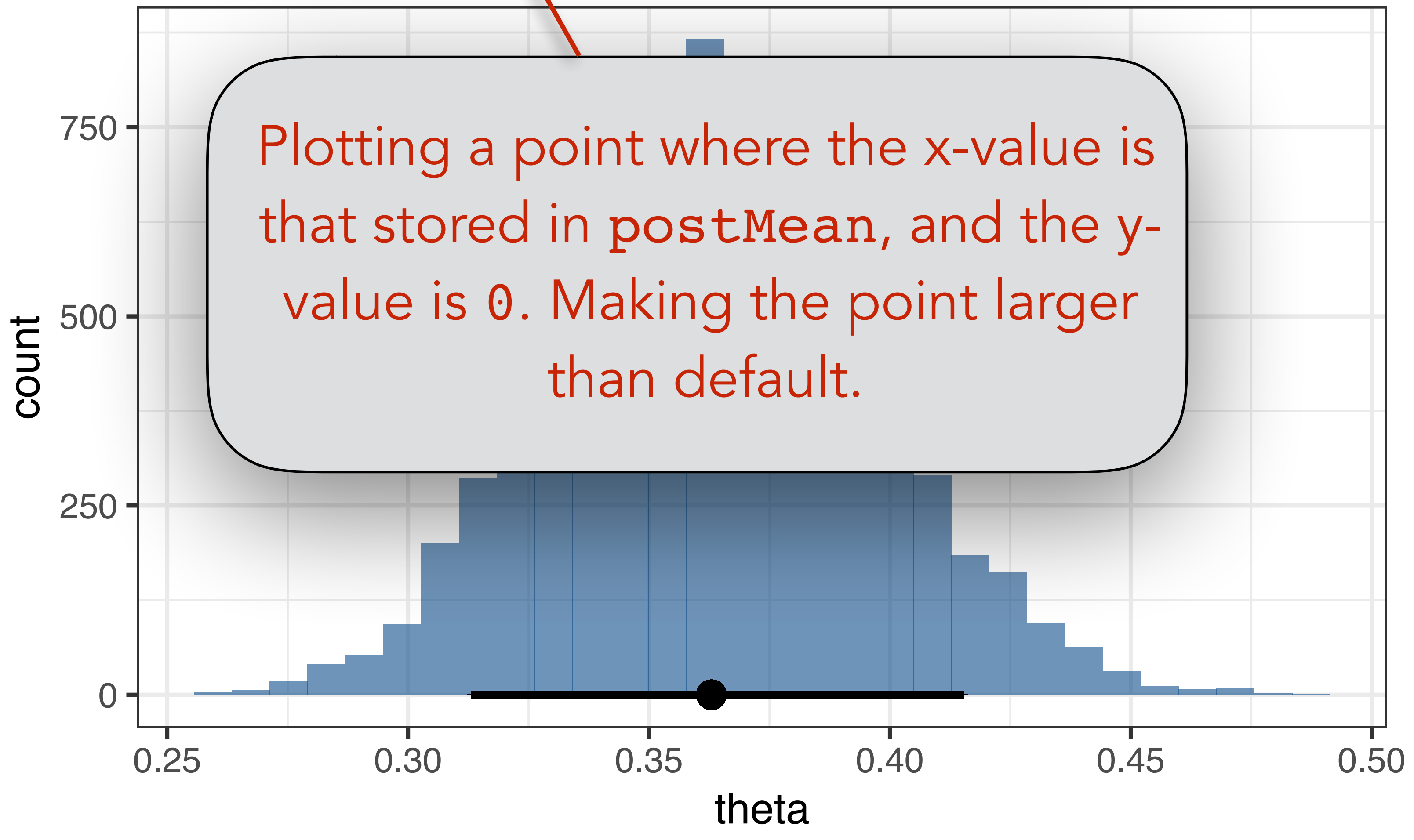


Plotting the histogram, customizing the fill colour and making it slightly transparent

5. Tentatively Interpret Results

Plot as a histogram

```
ggplot(posteriors, aes(x = theta)) +
  theme_bw() +
  geom_histogram(fill = "dodgerblue4", alpha = 0.6) +
  geom_point(aes(x = postMean, y = 0), size = 3) +
  geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
    size = 1)
```

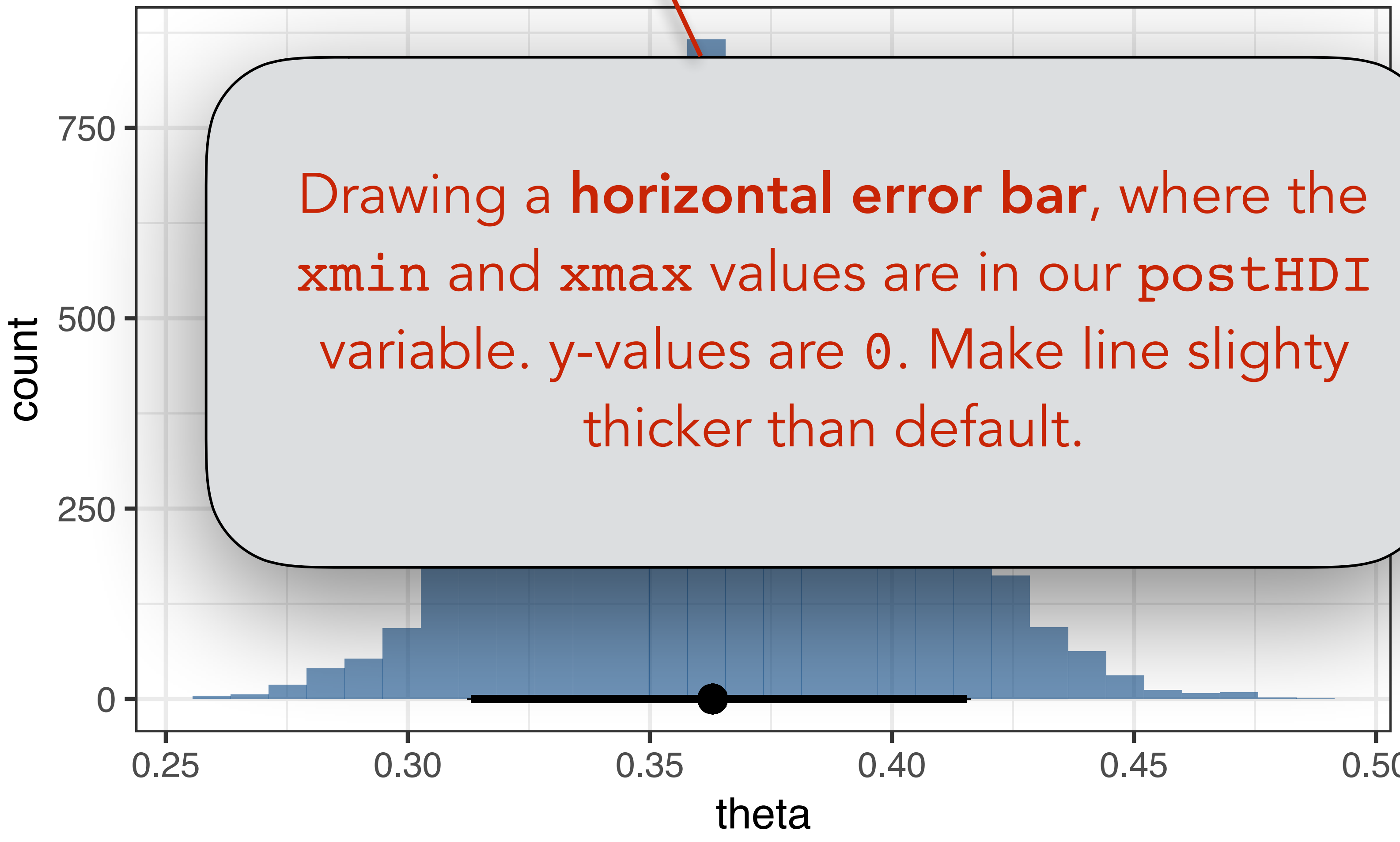


Plotting a point where the x-value is that stored in `postMean`, and the y-value is `0`. Making the point larger than default.

5. Tentatively Interpret Results

Plot as a histogram

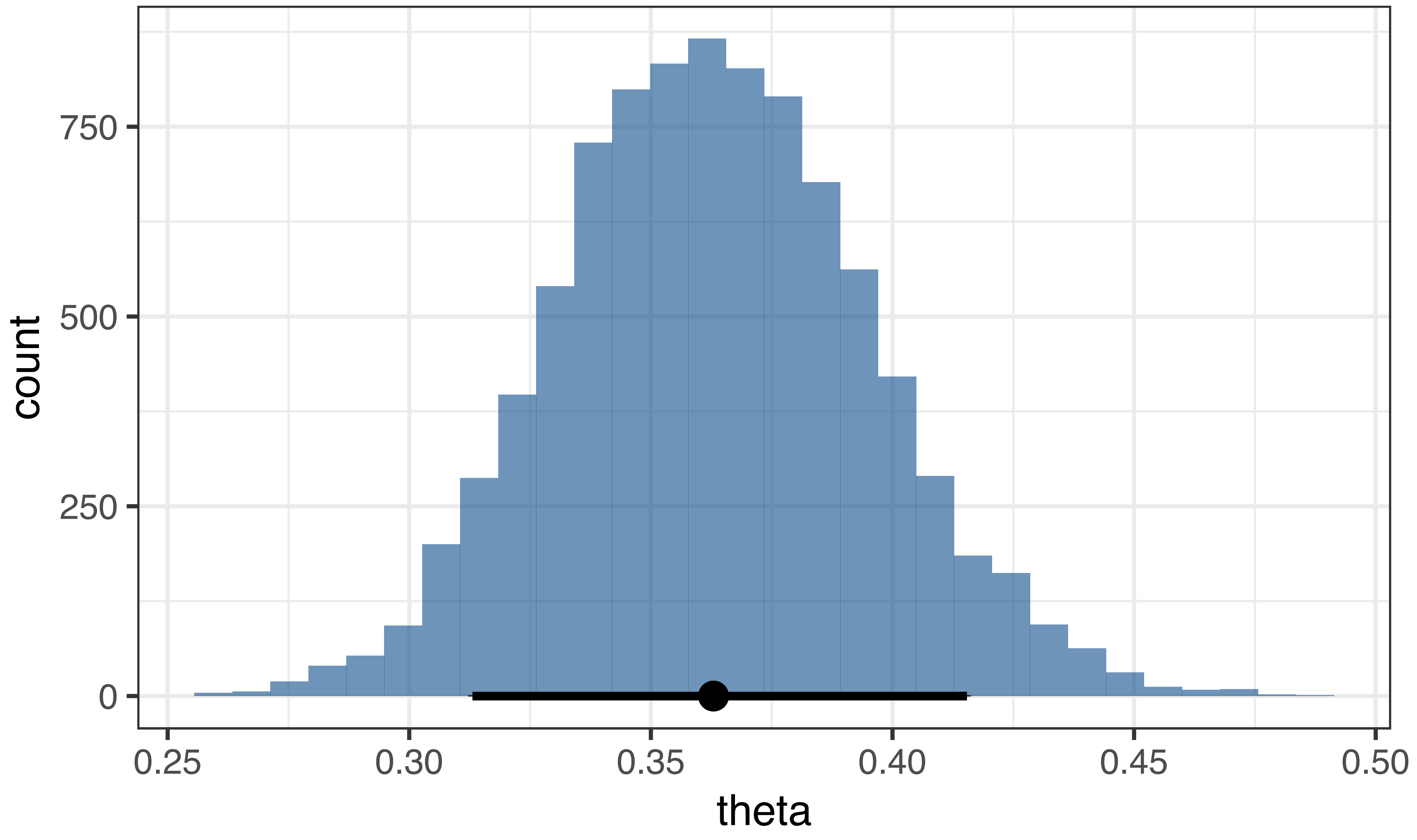
```
ggplot(posteriors, aes(x = theta)) +
  theme_bw() +
  geom_histogram(fill = "dodgerblue4", alpha = 0.6) +
  geom_point(aes(x = postMean, y = 0), size = 3) +
  geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
    size = 1)
```



Drawing a **horizontal error bar**, where the `xmin` and `xmax` values are in our `postHDI` variable. y-values are `0`. Make line slighty thicker than default.

5. Tentatively Interpret Results

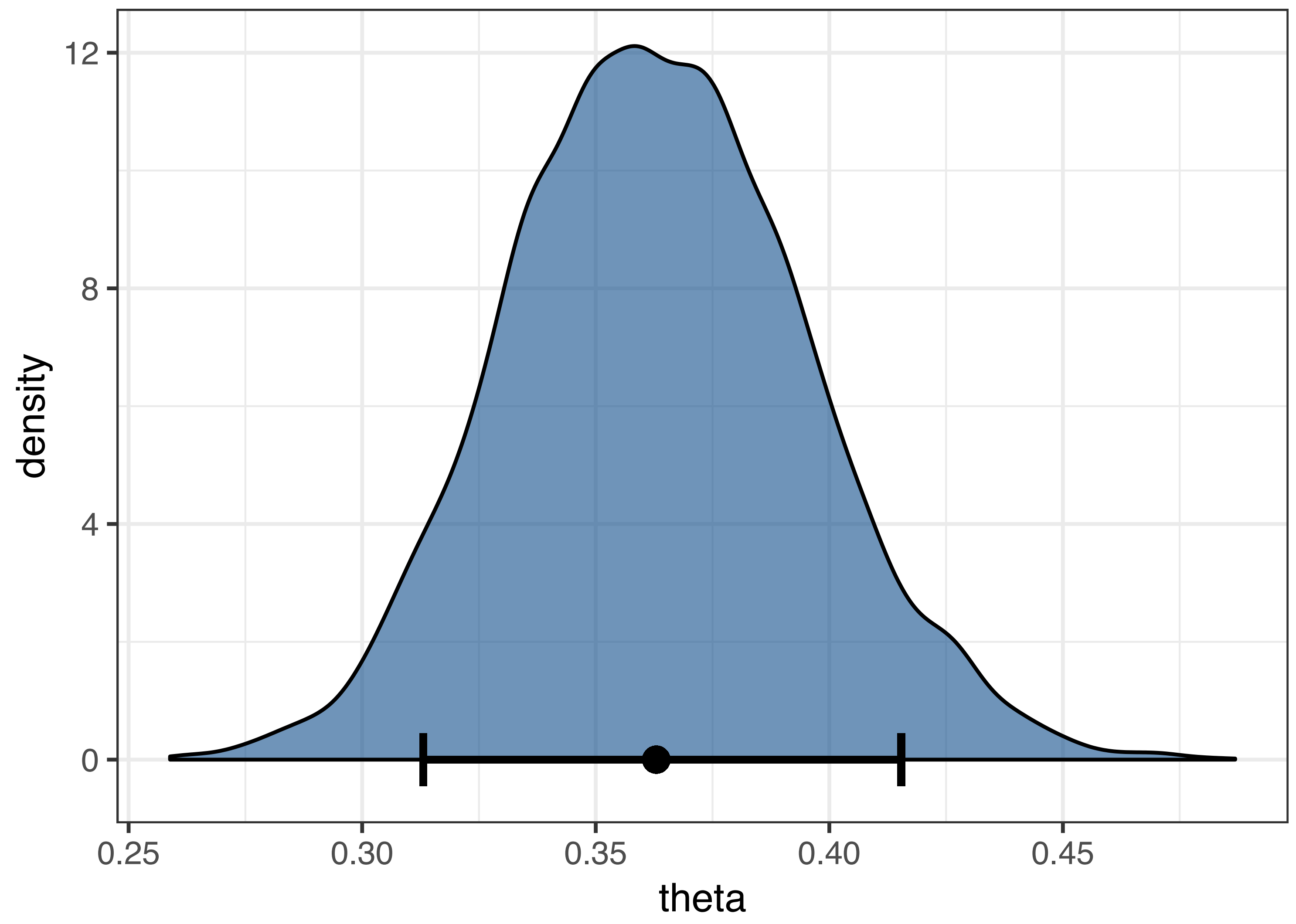
Plot as a histogram



5. Tentatively Interpret Results

Plot as a density plot

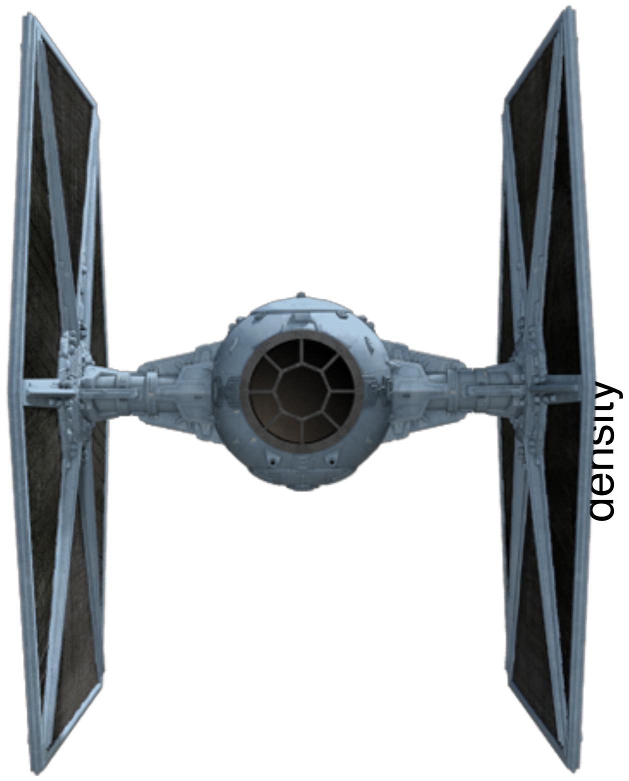
```
ggplot(posteriors, aes(x = theta)) +
  theme_bw() +
  geom_density(fill = "dodgerblue4", alpha = 0.6) +
  geom_point(aes(x = postMean, y = 0), size = 3) +
  geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
    size = 1)
```

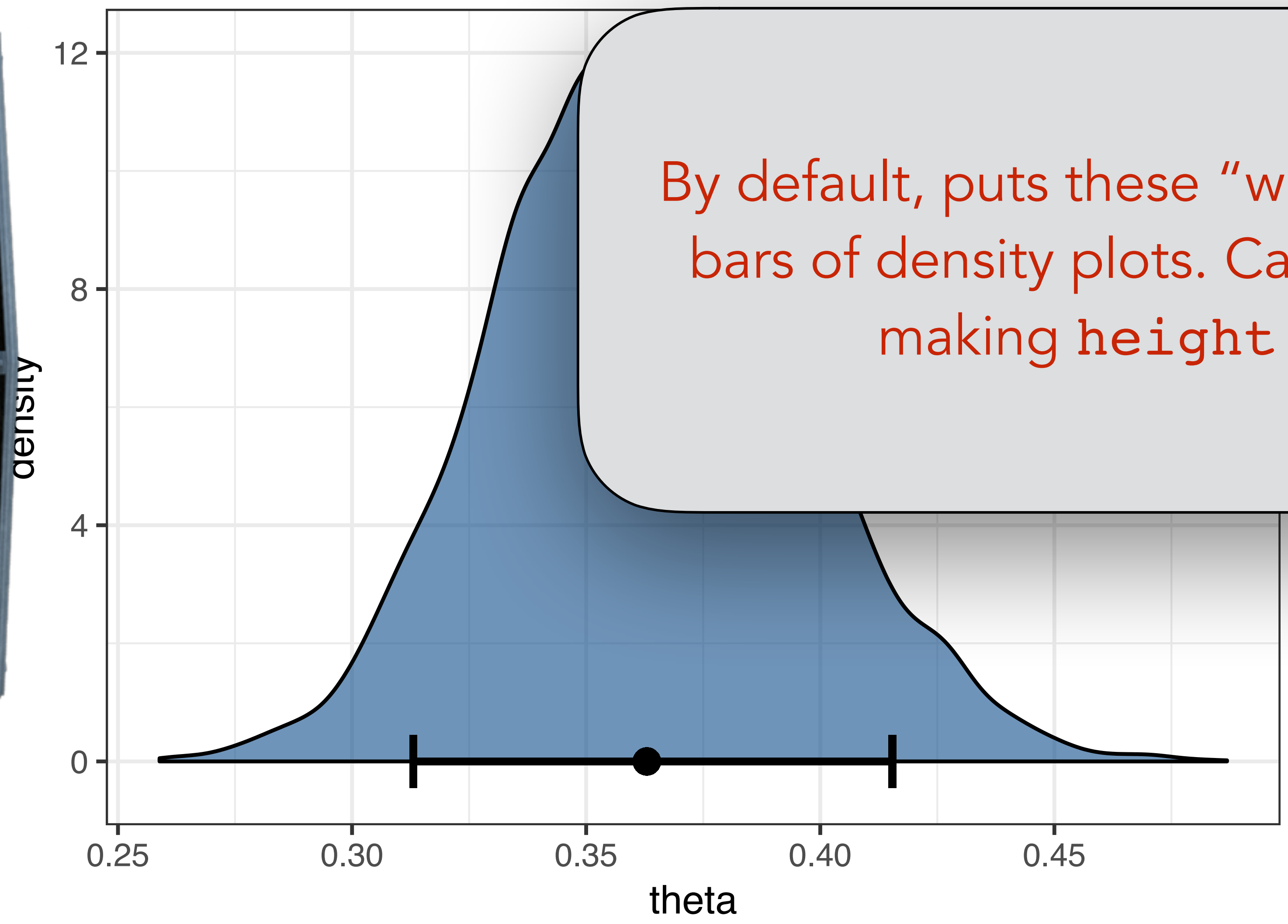


5. Tentatively Interpret Results

Plot as a density plot

```
ggplot(posteriors, aes(x = theta)) +
  theme_bw() +
  geom_density(fill = "dodgerblue4", alpha = 0.6) +
  geom_point(aes(x = postMean, y = 0), size = 3) +
  geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
    size = 1)
```



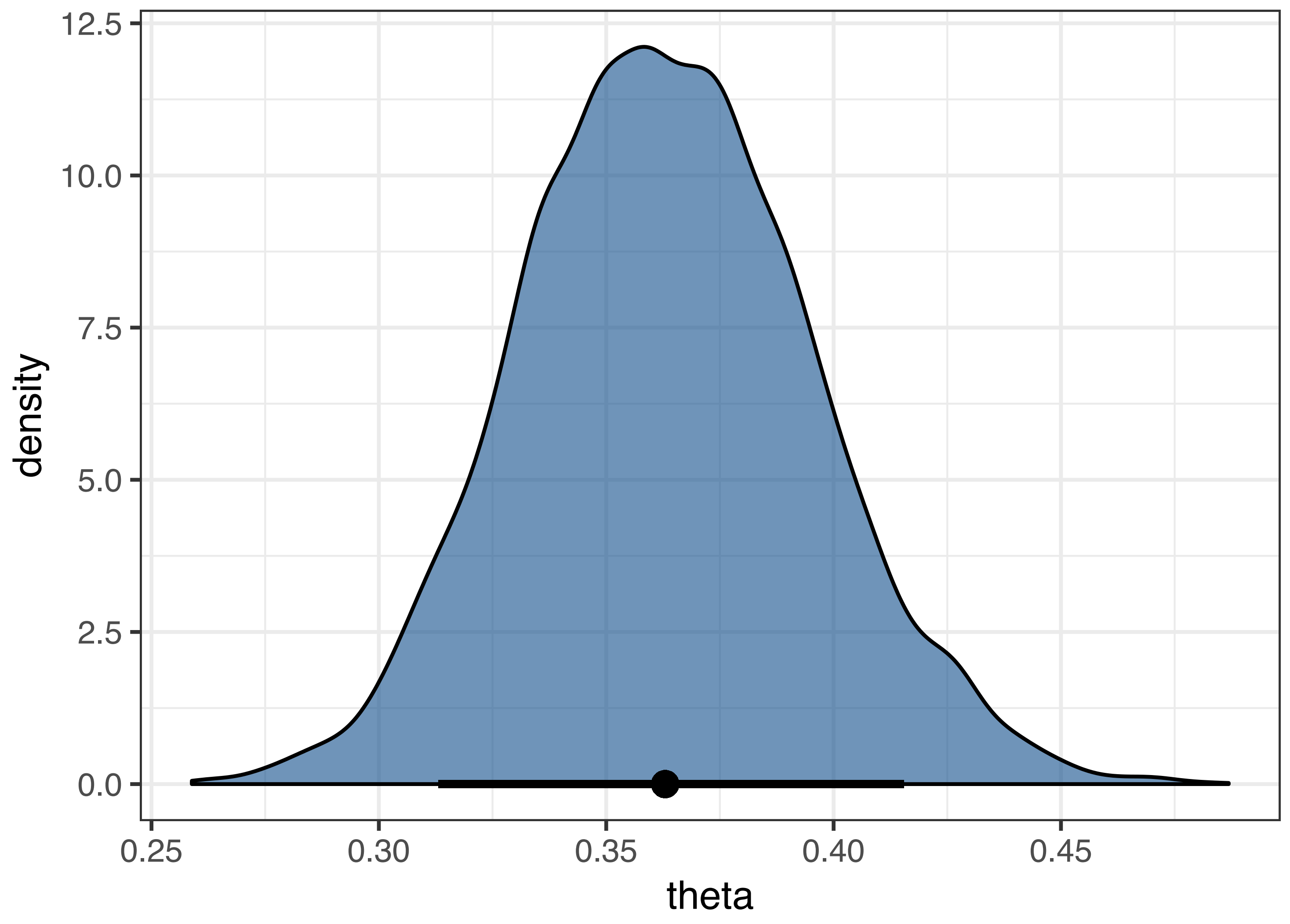


By default, puts these “wings” on error bars of density plots. Can change by making `height = 0`

5. Tentatively Interpret Results

Plot as a density plot

```
ggplot(posteriors, aes(x = theta)) +
  theme_bw() +
  geom_density(fill = "dodgerblue4", alpha = 0.6) +
  geom_point(aes(x = postMean, y = 0), size = 3) +
  geom_errorbarh(aes(y = 0, xmin = postHDI[1], xmax = postHDI[2]),
    size = 1, height = 0)
```



5. Tentatively Interpret Results

Plot as a point plot

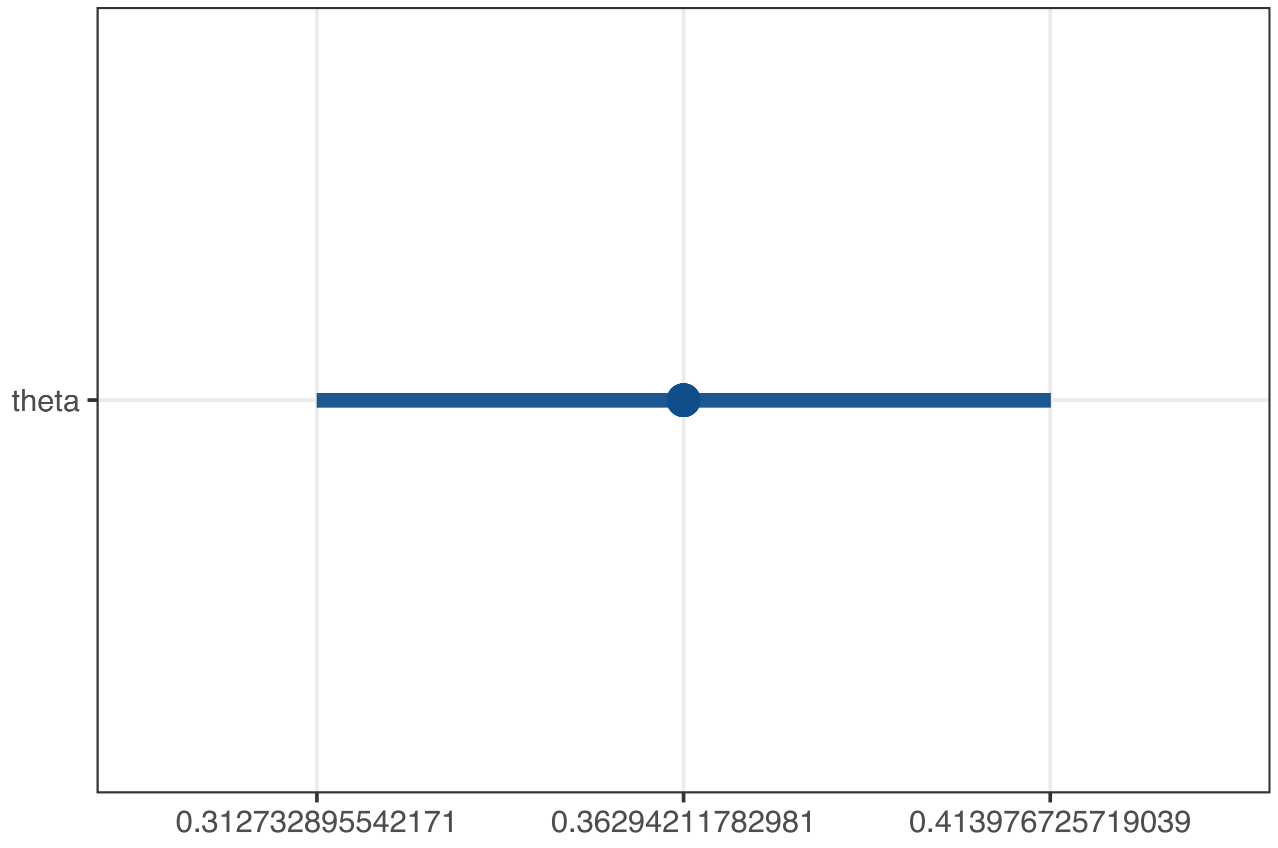
- Organize the data

```
data1 = c(postMean, postHDI)
data = data.frame(c("theta", data1))
data

  c..theta...data1.
1             theta
2  0.36294211782981
3 0.312732895542171
4 0.413976725719039
```

5. Tentatively Interpret Results
Plot as a point plot

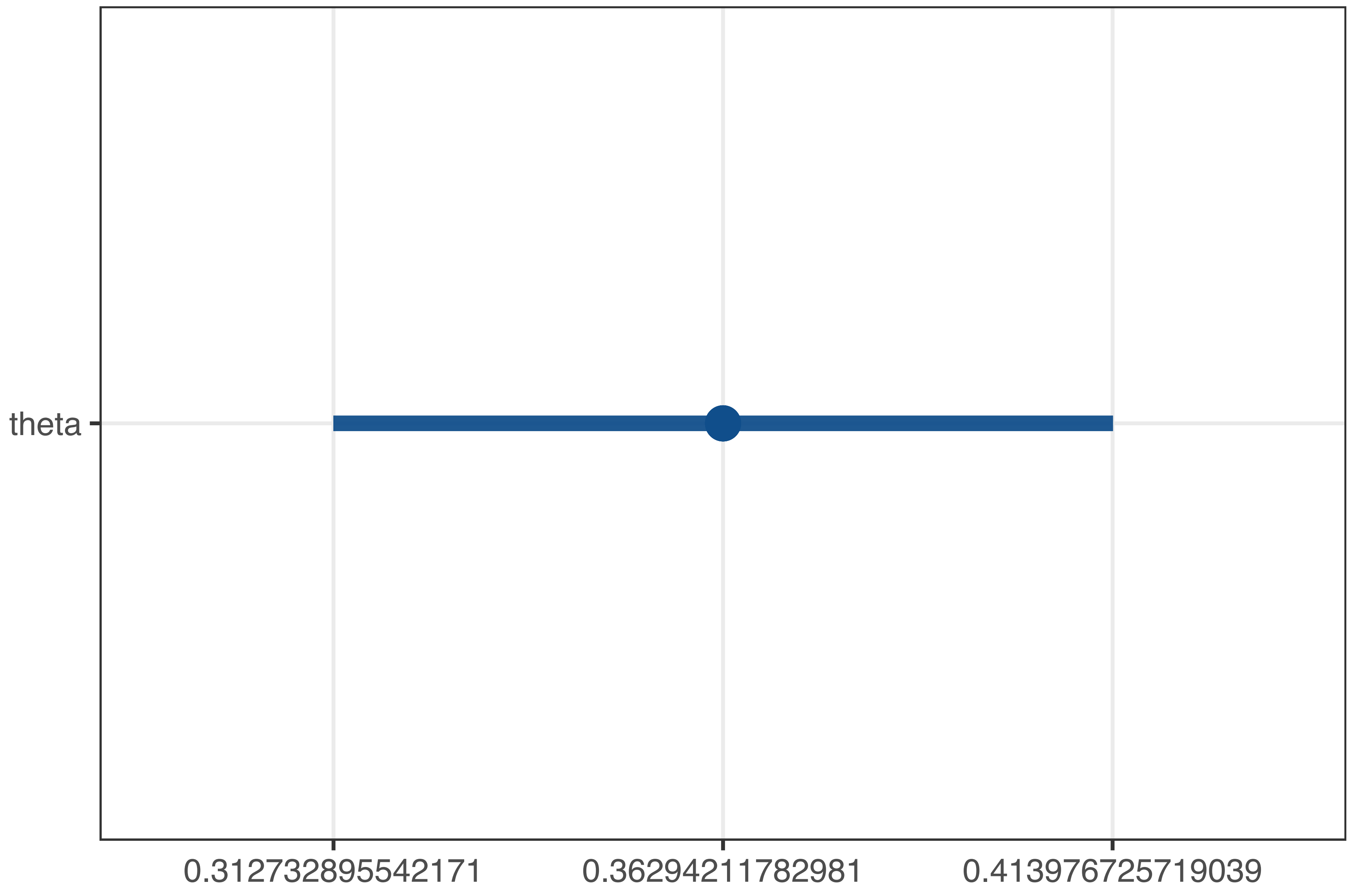
- Plot the data

```
ggplot(data, aes(y = data[1, 1])) +
  theme_bw() +
  geom_point(aes(x = data[2, 1]), size = 4, colour = "dodgerblue4") +
  geom_errorbarh(aes(xmin = data[3, 1], xmax = data[4, 1]), size = 2,
    colour = "dodgerblue4", alpha = 0.5, height = 0) +
  labs(y = "", x = "")
```



5. Tentatively Interpret Results

Plot as a point plot



5. Tentatively Interpret Results

Check validity of priors

- Can plot posteriors on top of priors for each parameter to see if priors might be having a large effect
- Must “regenerate” priors in R

5. Tentatively Interpret Results

Check validity of priors

- Generate prior values, same length as posterior

```
priors = runif(n = 9000, min = 0, max = 1)
```

5. Tentatively Interpret Results

Check validity of priors

- Generate prior values, same length as posterior

```
priors = runif(n = 9000, min = 0, max = 1)
```

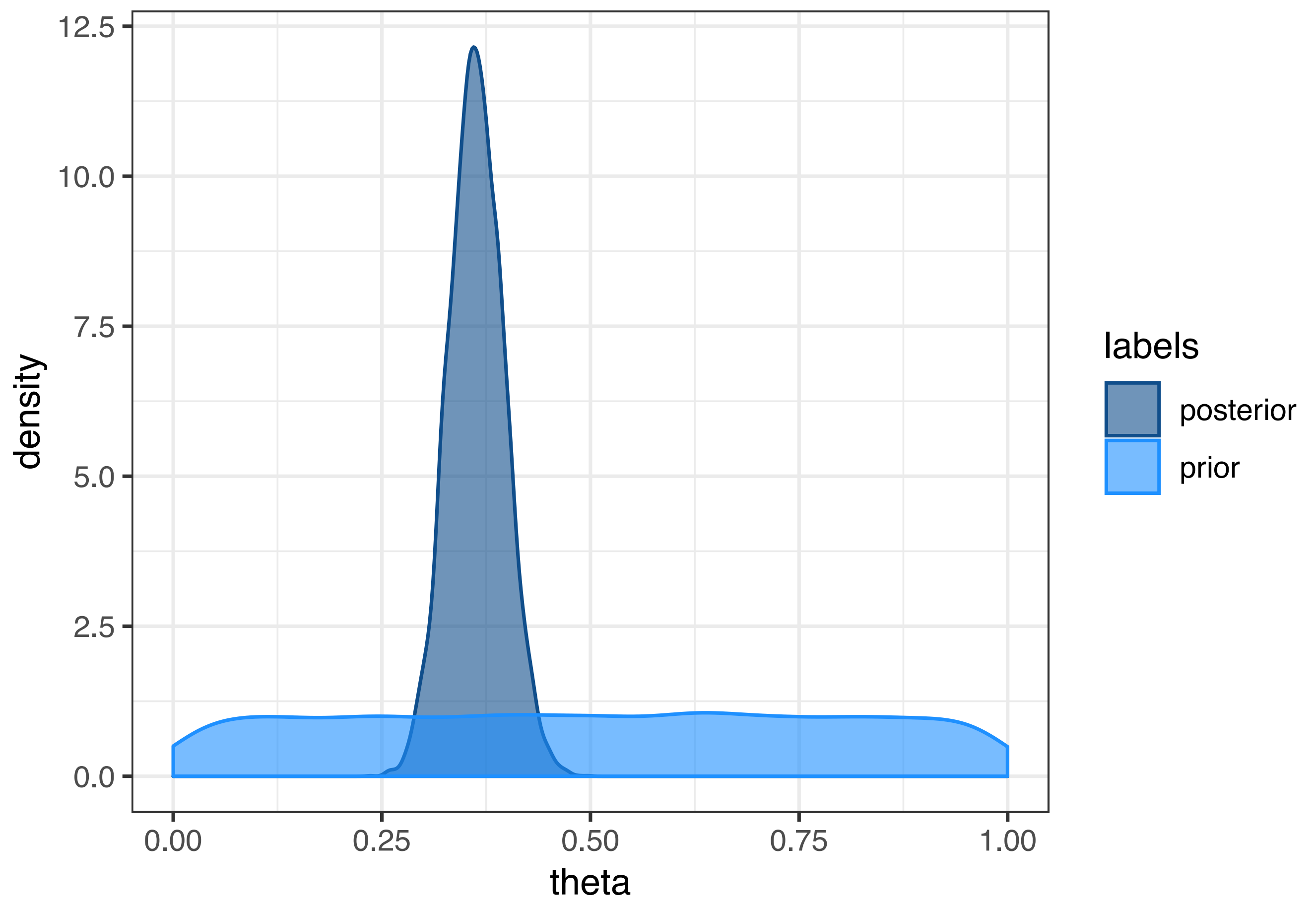
- Organize data into an appropriate data frame

```
labels = c(rep("prior", times = 9000), rep("posterior", times = 9000))
values = c(priors, posteriors$theta)
priorCheck = data.frame(labels, values)
head(priorCheck)

  labels    values
1  prior 0.9577340
2  prior 0.1657673
3  prior 0.7437207
4  prior 0.9777334
5  prior 0.3838118
6  prior 0.5193241
```

5. Tentatively Interpret Results

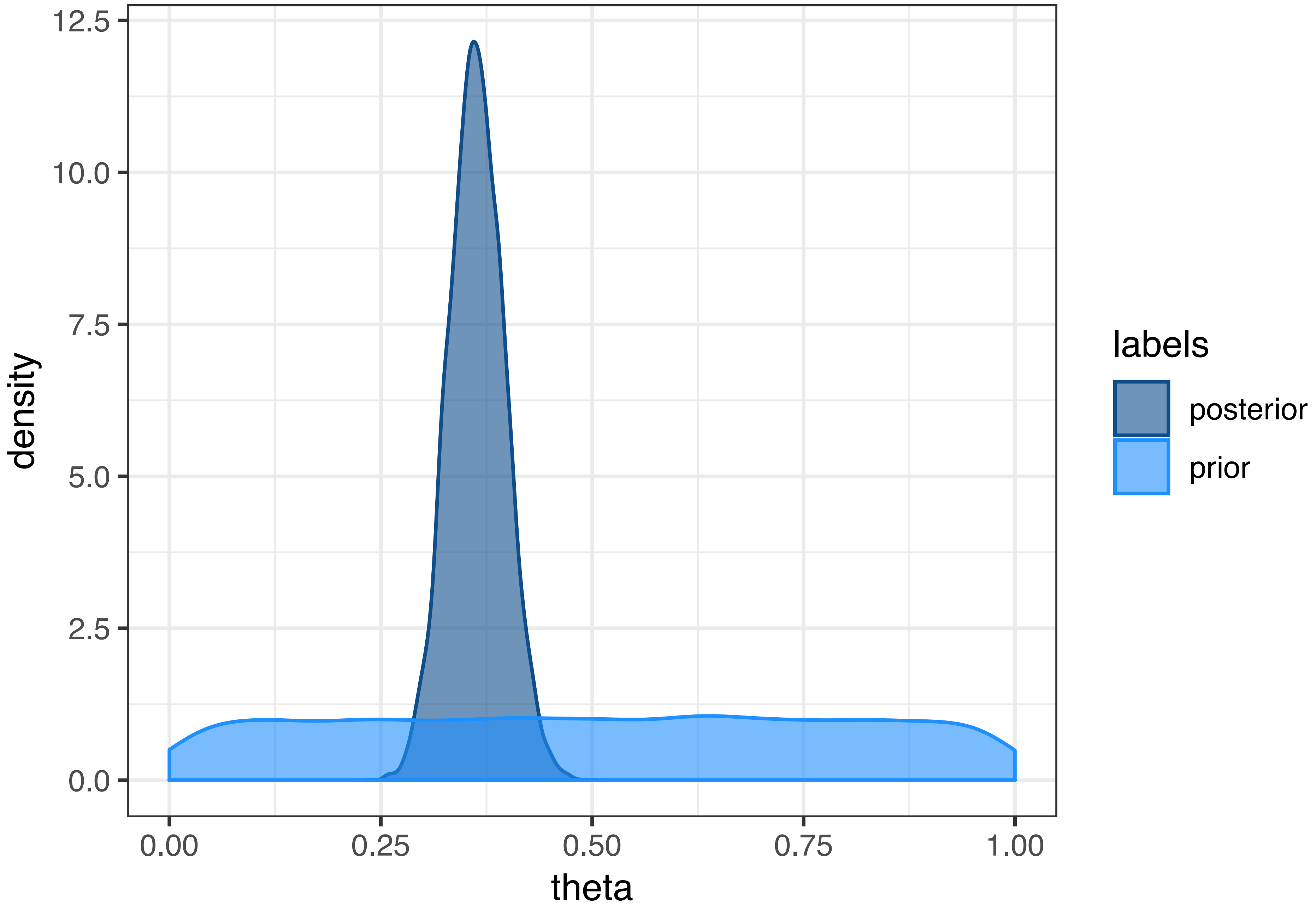
Check validity of priors

```
ggplot(priorCheck) +
  theme_bw() +
  geom_density(aes(x = values, group = labels, fill = labels, color =
    labels), alpha = 0.6) +
  scale_fill_manual(values = c("dodgerblue4", "dodgerblue")) +
  scale_color_manual(values = c("dodgerblue4", “dodgerblue")) +
  xlab(“theta”)
```



5. Tentatively Interpret Results

Check validity of priors



6. Conduct Posterior Predictive Checks

6. Conduct Posterior Predictive Checks

- Model seems to be behaving very well, but does it adequately account for processes (and their variation) resulting in data?

6. Conduct Posterior Predictive Checks

- Model seems to be behaving very well, but does it adequately account for processes (and their variation) resulting in data?
    - If so, generating random data from the model and parameter estimates should generate observed values
    - If not, should be a mismatch between generated values and actual data

6. Conduct Posterior Predictive Checks

- Stan allows for a specific “block” for doing just this (woohoo!)
  - Called `generated quantities`
- Add this to the model and re-run
  - Will include it from the beginning from here on out

6. Conduct Posterior Predictive Checks

```
generated quantities {
    real y_pred;
    y_pred = binomial_rng(N, theta);
  }
```

6. Conduct Posterior Predictive Checks

```
generated quantities {
    real y_pred;
    y_pred = binomial_rng(N, theta);
  }
```

Declare a new variable that will hold our results

6. Conduct Posterior Predictive Checks

```
generated quantities {
    real y_pred;
    y_pred = binomial_rng(N, theta);
  }
```

`rng` stands for "random number generator". All distributions have them in Stan.

Also note = rather than ~.

6. Conduct Posterior Predictive Checks

```
generated quantities {
    real y_pred;
    y_pred = binomial_rng(N, theta);
  }
```

For each step in the chain it will randomly sample a binomial distribution of size `N`, with probability `theta` (taken from the value for that step in the MCMC chain)

6. Conduct Posterior Predictive Checks

```
modelString = "
  data {
    int<lower=0> N;
    int<lower=0> y;
  }

  parameters {
    real theta;
  }

  model {
    // Priors
    theta ~ uniform(0, 1);

    // Likelihood
    y ~ binomial(N, theta);
  }

  generated quantities {
    real y_pred;
    y_pred = binomial_rng(N, theta);
  }
"
writeLines(modelString, con="model.stan")
```

6. Conduct Posterior Predictive Checks

```
modelString = "
  data {
    int<lower=0> N;
    int<lower=0> y;
  }

  parameters {
    real theta;
  }

  model {
    // Priors
    theta ~ uniform(0, 1);

    // Likelihood
    y ~ binomial(N, theta);
  }

  generated quantities {
    real y_pred;
    y_pred = binomial_rng(N, theta);
  }
"
writeLines(modelString, con="model.stan")
```

Will be almost identical to the likelihood.

6. Conduct Posterior Predictive Checks

- Run Stan

```
stanFit <- stan(file = "model.stan",
                data = dataList,
                pars = c("theta", "y_pred"),
                warmup = 2000,
                iter = 5000,
                chains = 3)
```

6. Conduct Posterior Predictive Checks

- Run Stan

```
stanFit <- stan(file = "model.stan",
                data = dataList,
                pars = c("theta", "y_pred"),
                warmup = 2000,
                iter = 5000,
                chains = 3)
```

Remember to tell Stan to keep track of the `y_pred` parameter!!!!

6. Conduct Posterior Predictive Checks

- Extract the data and calculate summary info

```
posteriors = as.data.frame(stanFit)
postMean = mean(posteriors$y_pred)
limits = quantile(posteriors$y_pred, probs = c(0.055, 0.945))
postHDI = as.numeric(limits)
```

6. Conduct Posterior Predictive Checks

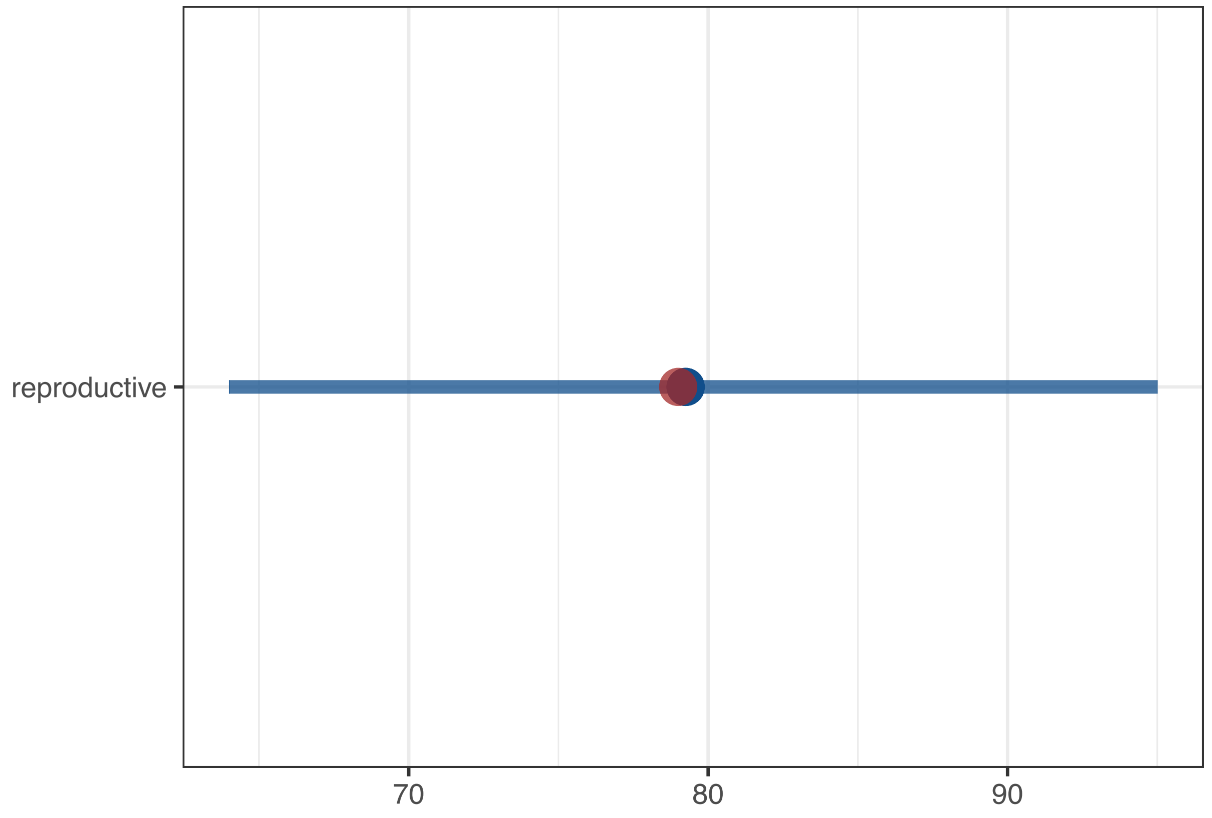
- Organize the data for plotting observed and expected results

```
data1 = data.frame(postMean, postHDI)
data2 = rep("reproductive", times = 2)
data3 = rep(mothers, times = 2)
data = data.frame(data2, data1, data3)
data

         data2 postMean postHDI data3
1 reproductive   79.249      64    79
2 reproductive   79.249      95    79
```

6. Conduct Posterior Predictive Checks

- Plot observed and expected results

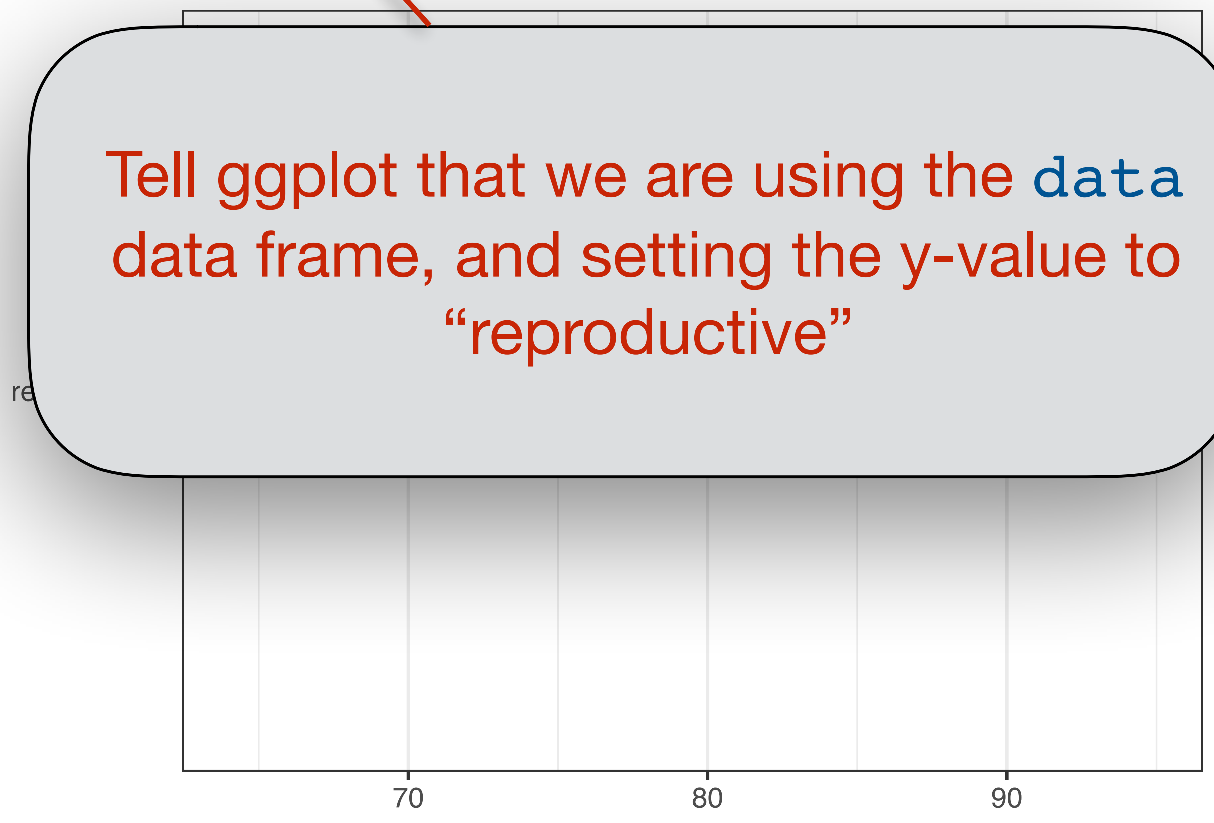
```
ggplot(data, aes(y = data[, 1])) +
  theme_bw() +
  geom_point(aes(x = data[, 2]), size = 5, colour = "dodgerblue4") +
  geom_errorbarh(aes(xmin = data[1, 3], xmax = data[2, 3]), size = 2,
    colour = "dodgerblue4", alpha = 0.5, height = 0) +
  geom_point(aes(x = data[, 4]), size = 5, colour = "brown", alpha = 0.5) +
  labs(y = "", x = "")
```



6. Conduct Posterior Predictive Checks

- Plot observed and expected results

```
ggplot(data, aes(y = data[, 1])) +
    theme_bw() +
    geom_point(aes(x = data[, 2]), size = 5, colour = "dodgerblue4") +
    geom_errorbarh(aes(xmin = data[1, 3], xmax = data[2, 3]), size = 2,
      colour = "dodgerblue4", alpha = 0.5, height = 0) +
    geom_point(aes(x = data[, 4]), size = 5, colour = "brown", alpha = 0.5) +
    labs(y = "", x = "")
```

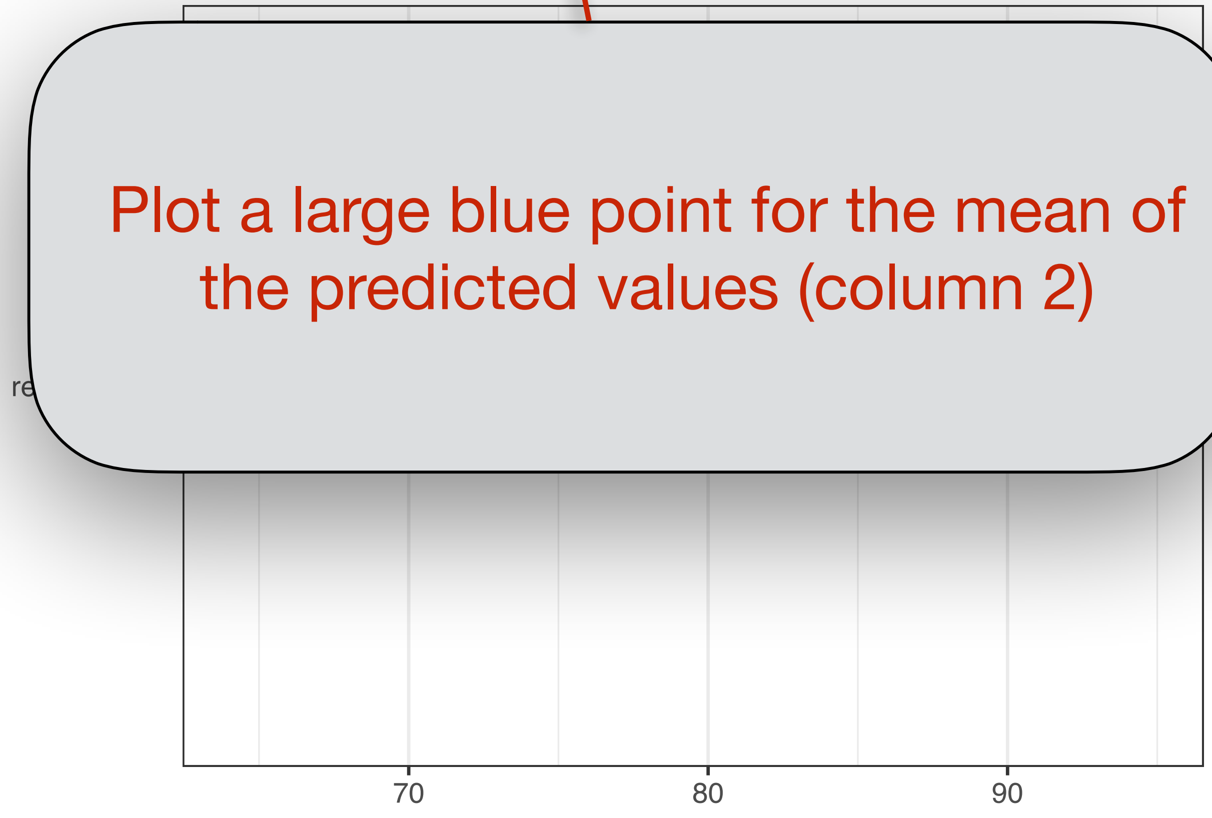


Tell ggplot that we are using the `data` data frame, and setting the y-value to "reproductive"

6. Conduct Posterior Predictive Checks

- Plot observed and expected results

```
ggplot(data, aes(y = data[, 1])) +
  theme_bw() +
  geom_point(aes(x = data[, 2]), size = 5, colour = "dodgerblue4") +
  geom_errorbarh(aes(xmin = data[1, 3], xmax = data[2, 3]), size = 2,
    colour = "dodgerblue4", alpha = 0.5, height = 0) +
  geom_point(aes(x = data[, 4]), size = 5, colour = "brown", alpha = 0.5) +
  labs(y = "", x = "")
```

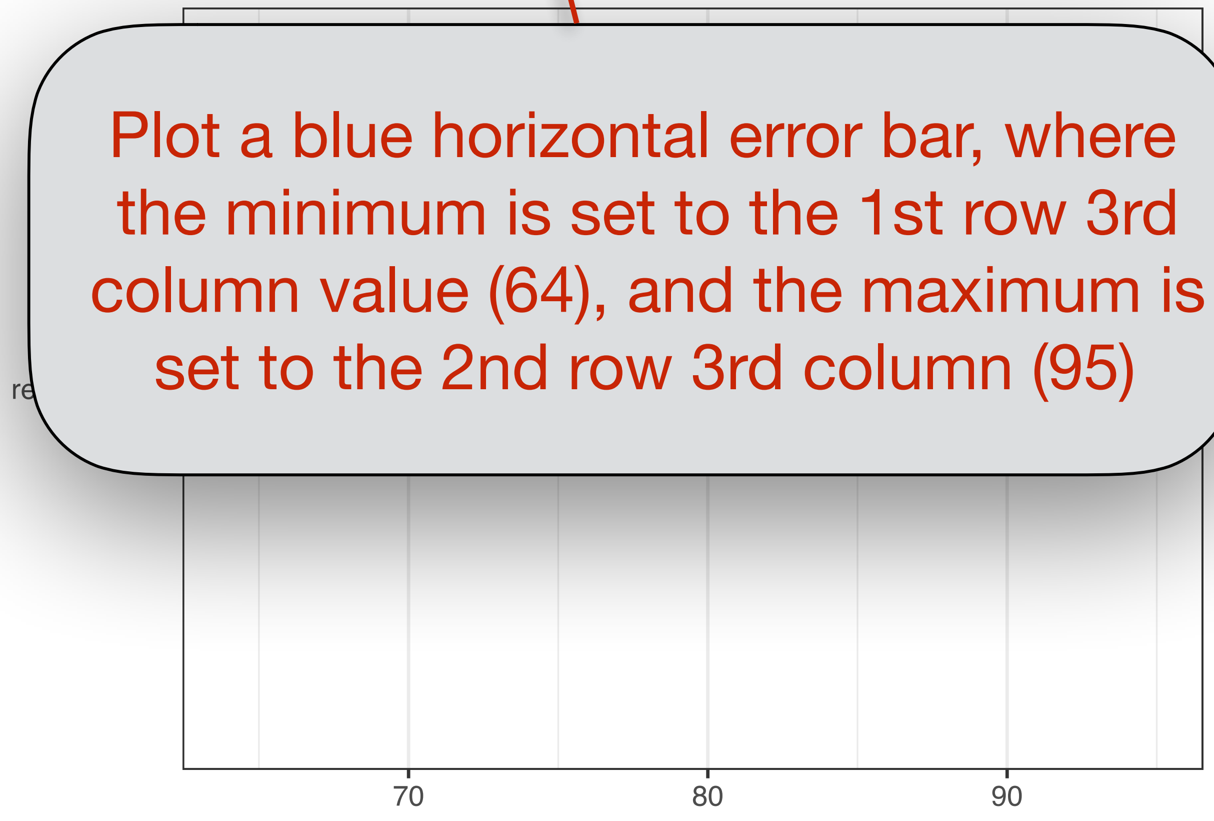


Plot a large blue point for the mean of the predicted values (column 2)

6. Conduct Posterior Predictive Checks

- Plot observed and expected results

```
ggplot(data, aes(y = data[, 1])) +
  theme_bw() +
  geom_point(aes(x = data[, 2]), size = 5, colour = "dodgerblue4") +
  geom_errorbarh(aes(xmin = data[1, 3], xmax = data[2, 3]), size = 2,
    colour = "dodgerblue4", alpha = 0.5, height = 0) +
  geom_point(aes(x = data[, 4]), size = 5, colour = "brown", alpha = 0.5) +
  labs(y = "", x = "")
```

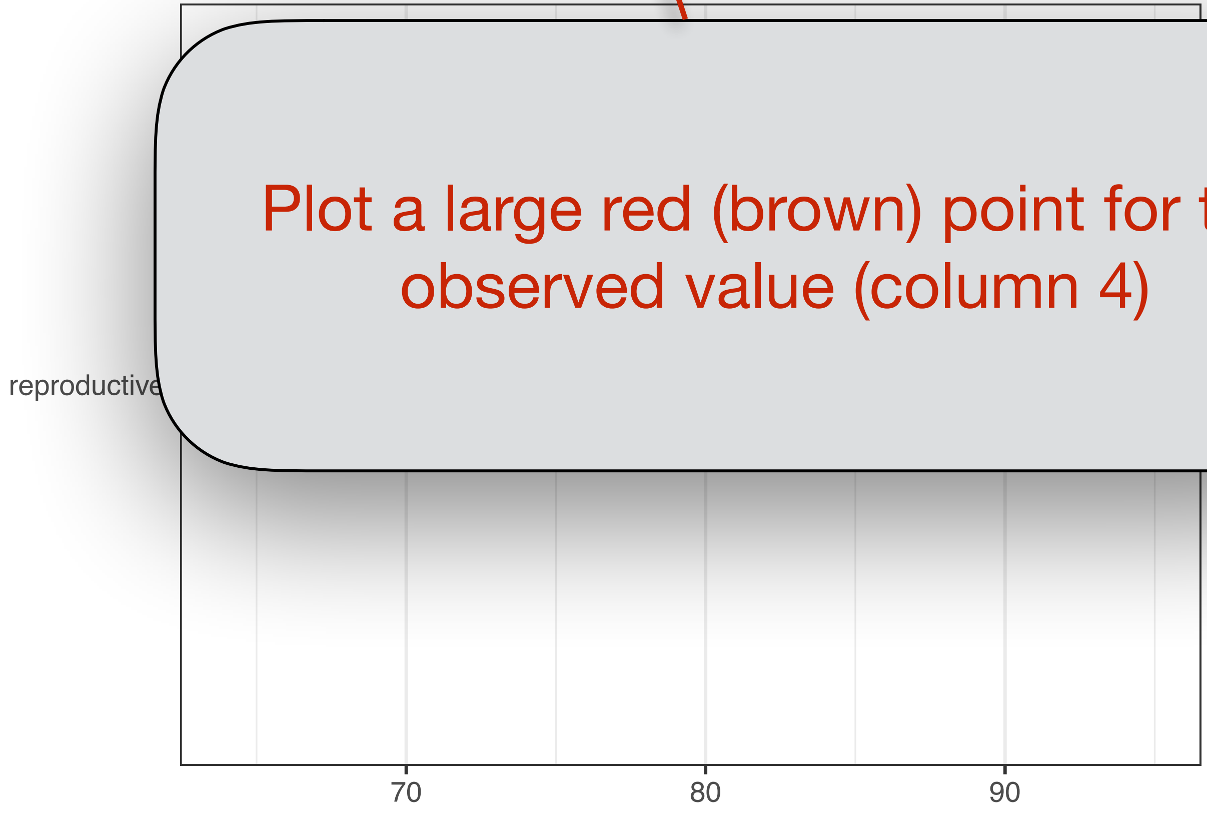


Plot a blue horizontal error bar, where the minimum is set to the 1st row 3rd column value (64), and the maximum is set to the 2nd row 3rd column (95)

6. Conduct Posterior Predictive Checks

- Plot observed and expected results

```
ggplot(data, aes(y = data[, 1])) +
  theme_bw() +
  geom_point(aes(x = data[, 2]), size = 5, colour = "dodgerblue4") +
  geom_errorbarh(aes(xmin = data[1, 3], xmax = data[2, 3]), size = 2,
    colour = "dodgerblue4", alpha = 0.5, height = 0) +
  geom_point(aes(x = data[, 4]), size = 5, colour = "brown", alpha = 0.5) +
  labs(y = "", x = "")
```



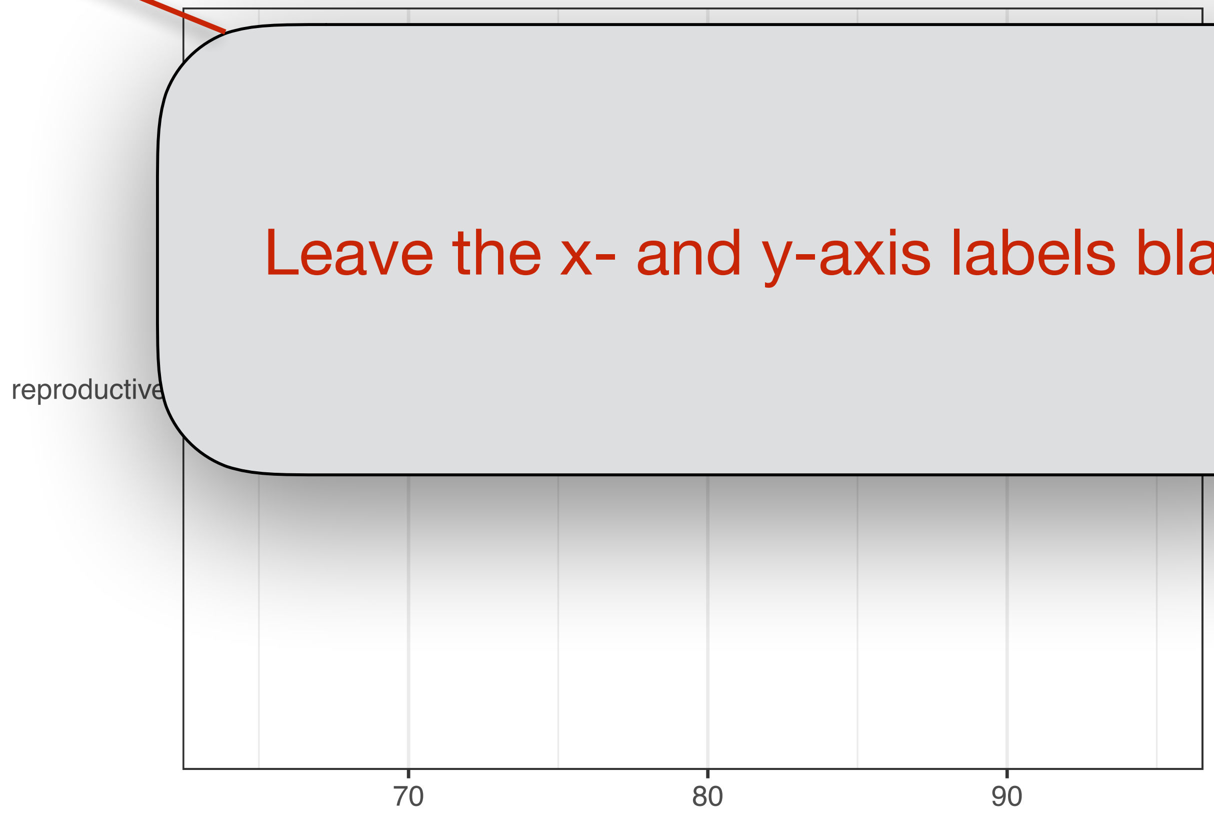
Plot a large red (brown) point for the observed value (column 4)

6. Conduct Posterior Predictive Checks

- Plot observed and expected results

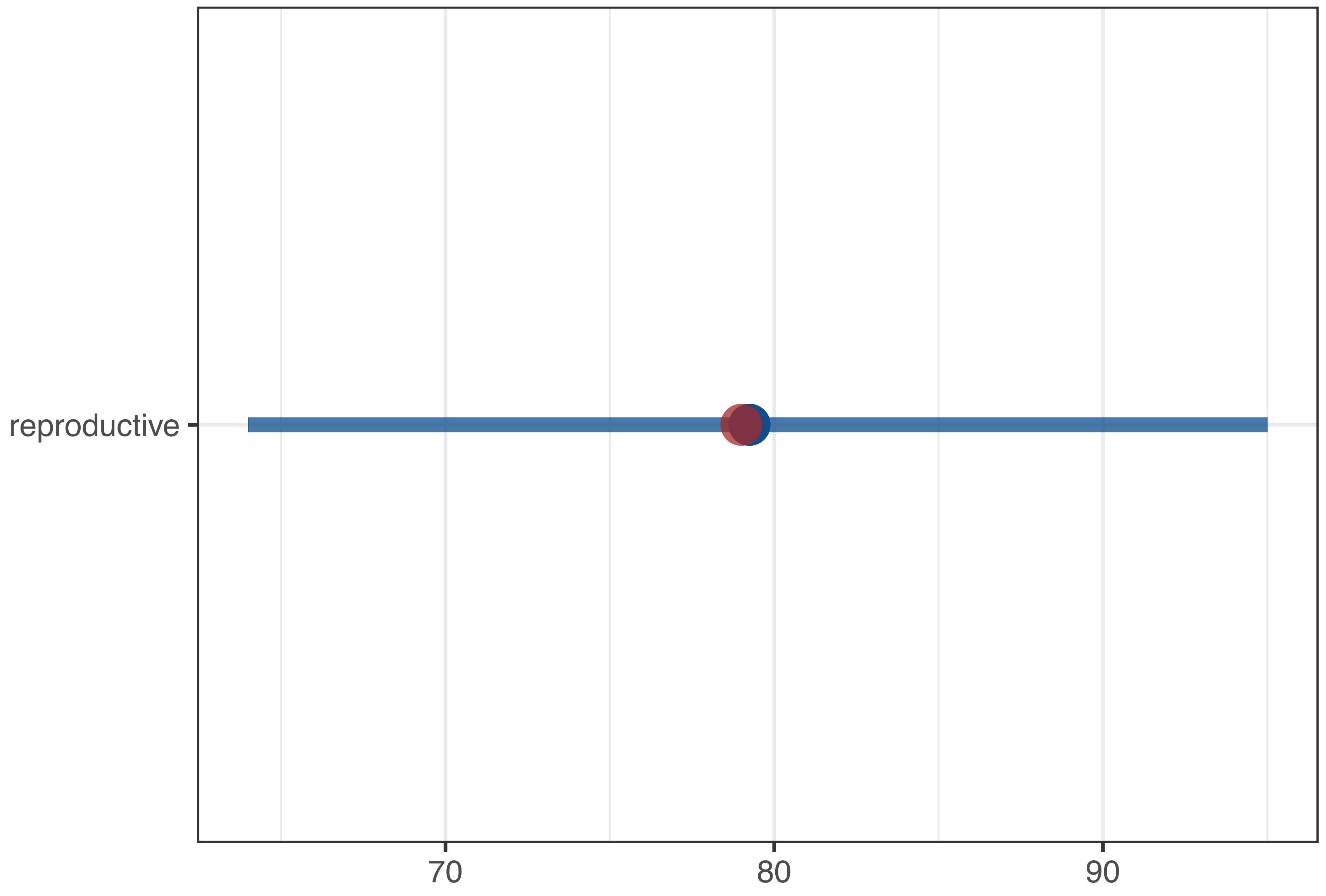
```
ggplot(data, aes(y = data[, 1])) +
  theme_bw() +
  geom_point(aes(x = data[, 2]), size = 5, colour = "dodgerblue4") +
  geom_errorbarh(aes(xmin = data[1, 3], xmax = data[2, 3]), size = 2,
    colour = "dodgerblue4", alpha = 0.5, height = 0) +
  geom_point(aes(x = data[, 4]), size = 5, colour = "brown", alpha = 0.5) +
  labs(y = "", x = "")
```

Leave the x- and y-axis labels blank



6. Conduct Posterior Predictive Checks

- Looks pretty bang on!!!



Questions?