Bayes!

Frank Edwards

2/24/2021

Interpret these results

```
library(broom)
data(mtcars)
## mpg is miles per gallon, wt is weight in 1000 lbs
m1<-lm(mpg ~ wt, data = mtcars)
tidy(m1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    37.3      1.88      19.9  8.24e-19
## 2 wt             -5.34     0.559     -9.56 1.29e-10
```

Uncertainty in parameter estimates: frequentist

```
### The Frequentist confidence interval: interpret
confint(m1)
```

```
##                 2.5 %    97.5 %
## (Intercept) 33.450500 41.119753
## wt          -6.486308 -4.202635
```

Motivating two common approaches

- **Frequentist**: The truth is fixed, we can estimate it using repeated sampling and large number theorems, assuming our measurement is one of many that could have resulted

Motivating two common approaches

- **Frequentist**: The truth is fixed, we can estimate it using repeated sampling and large number theorems, assuming our measurement is one of many that could have resulted
- **Bayesian**: Given our assumptions and the data, which probability distribution is the most plausible answer?

Uncertainty in parameter estimates: Bayesian

```
### estimate the model as a Bayesian lm
m1_b<-stan_glm(mpg ~ wt, data = mtcars)

### format the posterior as a tibble
m1_b_post<-as_tibble(m1_b)

### what the heck is this
head(m1_b_post)
```

```
## # A tibble: 6 x 3
##   `(Intercept)`    wt sigma
##           <dbl> <dbl> <dbl>
## 1          37.9 -5.42  3.08
## 2          36.9 -5.35  3.55
## 3          36.6 -5.24  3.27
## 4          39.9 -5.96  3.47
## 5          34.5 -4.77  2.99
## 6          34.9 -4.80  3.07
```

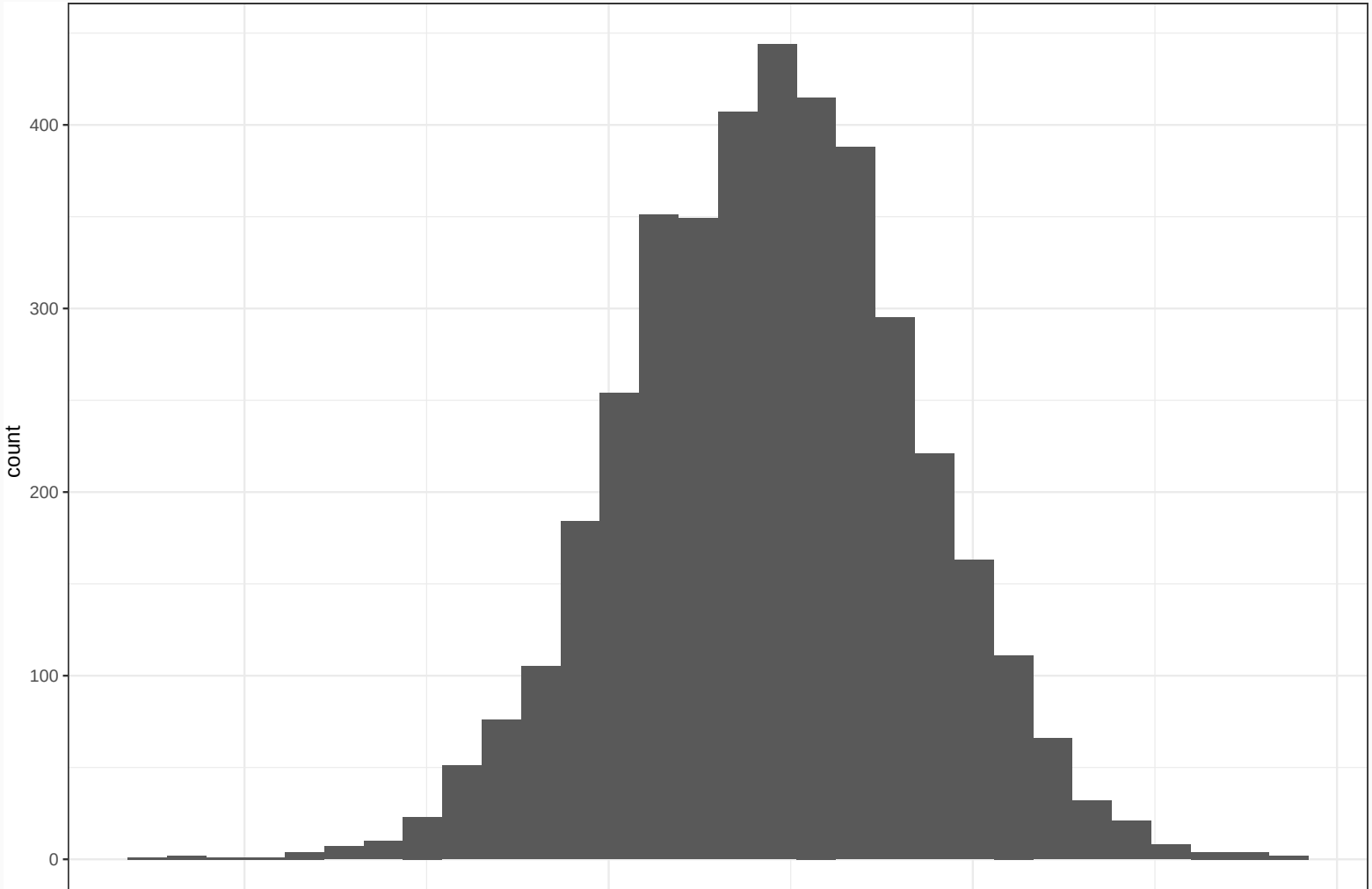
Using posterior draws

```
### now directly compute uncertainty interval
m1_b_post %>%
  summarize(intercept_md = median(`(Intercept)`),
            intercept_lwr = quantile(`(Intercept)`, 0.025),
            intercept_upr = quantile(`(Intercept)`, 0.975))
```

```
## # A tibble: 1 x 3
##   intercept_md intercept_lwr intercept_upr
##          <dbl>         <dbl>         <dbl>
## 1         37.2          33.3          41.1
```

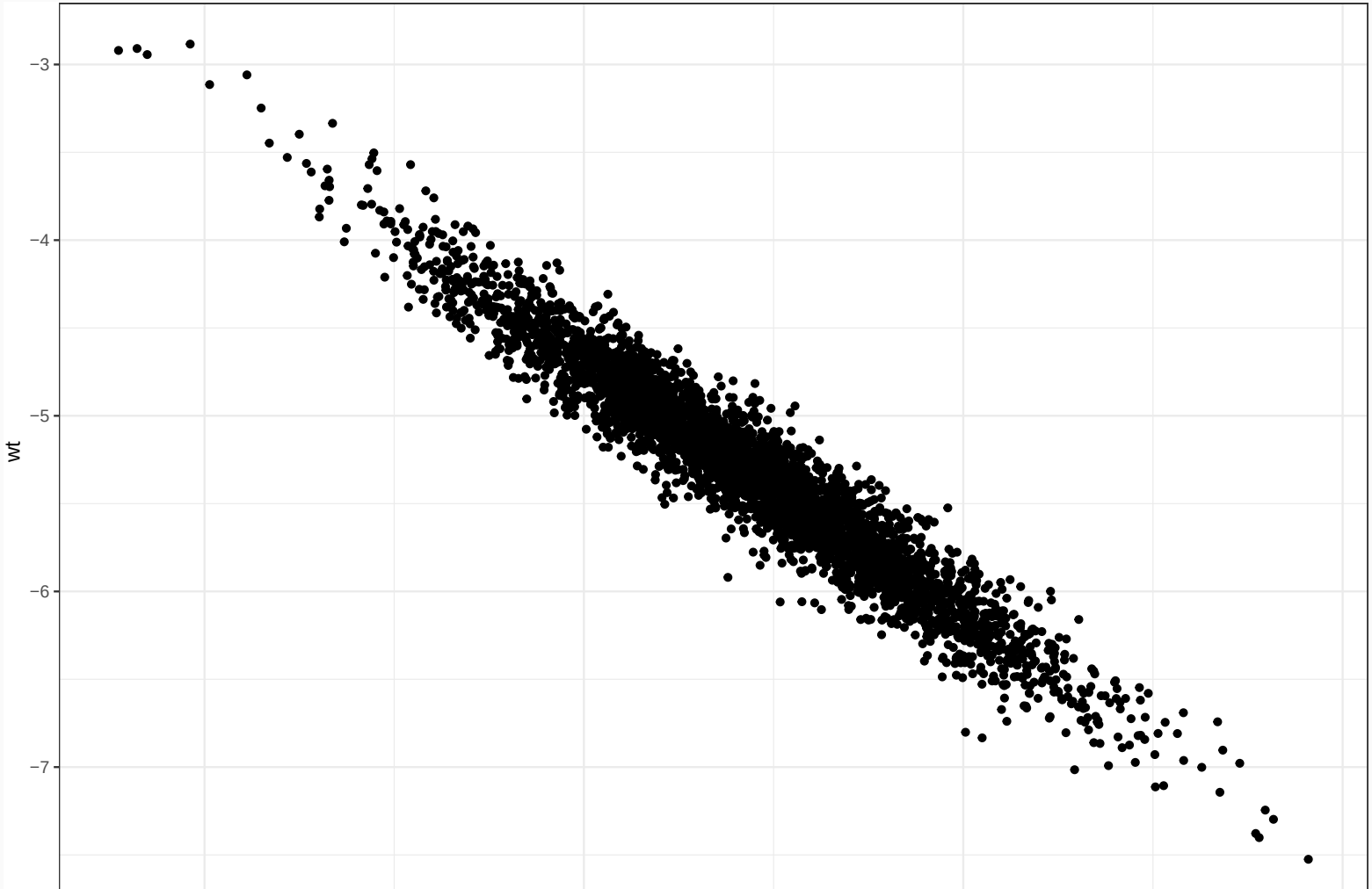
Using posterior draws

```
### or plot it!
ggplot(m1_b_post) +
  geom_histogram(aes(x = `(Intercept)`))
```



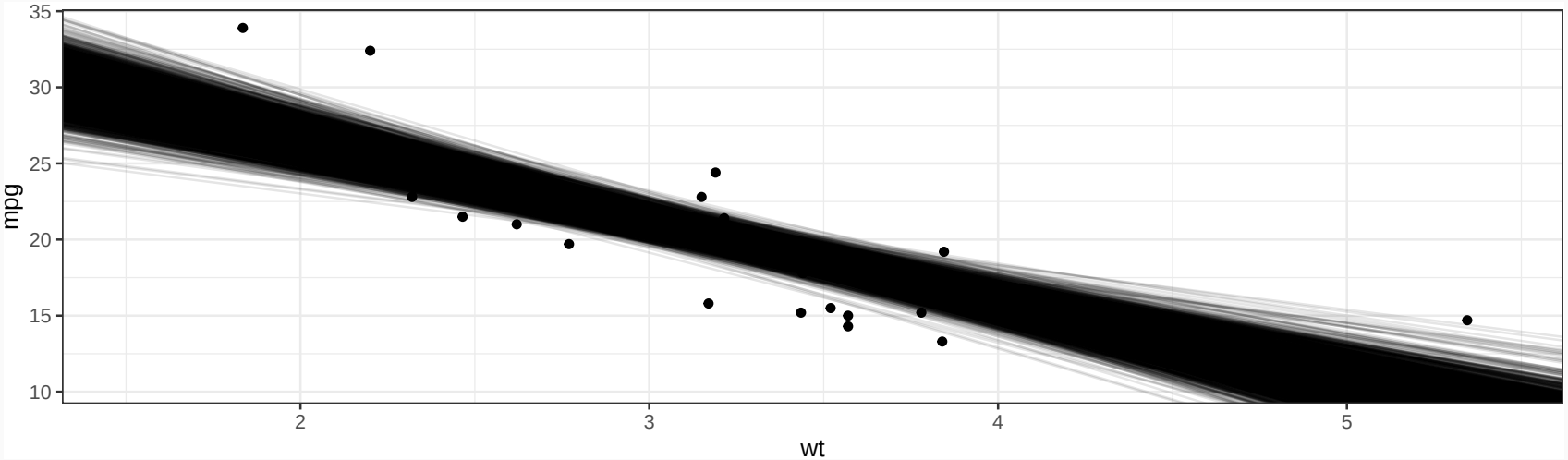
Using posterior draws

```
### or plot it!
ggplot(m1_b_post) +
  geom_point(aes(x = `(Intercept)`, y = wt))
```



Using posterior draws: fitted regression lines

```
### or plot it!
ggplot(mtcars, aes(y = mpg, x = wt)) +
  geom_point()+
  geom_abline(data = m1_b_post,
              aes(intercept = `(Intercept)`,
                  slope = wt),
              alpha = 0.1)
```



Some quick math: The likelihood and Maximum likelihood

The likelihood function is the probability density of the data, conditional on the parameters and predictors. Here, we use the Normal probability density function to model the likelihood of y. We routinely use a probability distribution to express the likelihood

$$p(y|X) \sim N(\mu, \sigma)$$

Some quick math: The likelihood and Maximum likelihood

The likelihood function is the probability density of the data, conditional on the parameters and predictors. Here, we use the Normal probability density function to model the likelihood of y. We routinely use a probability distribution to express the likelihood

$$p(y|X) \sim N(\mu, \sigma)$$

To estimate frequentist linear regression models, we can use **Maximum Likelihood Estimation**.

$$p(y|a, b, \sigma, X) = \prod_{i=1}^{n} \text{Normal}(y_i|a + bx_i, \sigma^2)$$

Some quick math: The likelihood and Maximum likelihood

The likelihood function is the probability density of the data, conditional on the parameters and predictors. Here, we use the Normal probability density function to model the likelihood of y. We routinely use a probability distribution to express the likelihood

$$p(y|X) \sim N(\mu, \sigma)$$

To estimate frequentist linear regression models, we can use **Maximum Likelihood Estimation**.

$$p(y|a, b, \sigma, X) = \prod_{i=1}^{n} \text{Normal}(y_i|a + bx_i, \sigma^2)$$

We can solve for the maximum of the likelihood equation by taking the product of all Normal PDF values at each data point. The most likely parameter values are those values where the likelihood is maximized.

Some quick math: Bayesian model estimation

Bayes' rule

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

Some quick math: Bayesian model estimation

Bayes' rule

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

Bayes' rule: model form

$$p(\text{parameter}|\text{Data}) = \frac{p(\text{Data}|\text{parameter}) \cdot p(\text{parameter})}{p(\text{Data})}$$

Translating this further

Bayes' rule: model form

$$p(\text{parameter}|\text{Data}) = \frac{p(\text{Data}|\text{parameter}) \cdot p(\text{parameter})}{p(\text{Data})}$$

Translating this further

Bayes' rule: model form

$$p(\text{parameter}|\text{Data}) = \frac{p(\text{Data}|\text{parameter}) \cdot p(\text{parameter})}{p(\text{Data})}$$

$$\text{posterior} \propto \text{likelihood} \cdot \text{prior}$$

An example

Let’s flip coins!

We are going to flip 15 coins, but have no idea how likely this coin is to land on heads.

Let's flip coins!

We are going to flip 15 coins, but have no idea how likely this coin is to land on heads.

Likelihood

$$y_i \sim Binomial(15, p)$$

Let's flip coins!

We are going to flip 15 coins, but have no idea how likely this coin is to land on heads.

Likelihood

$$y_i \sim Binomial(15, p)$$

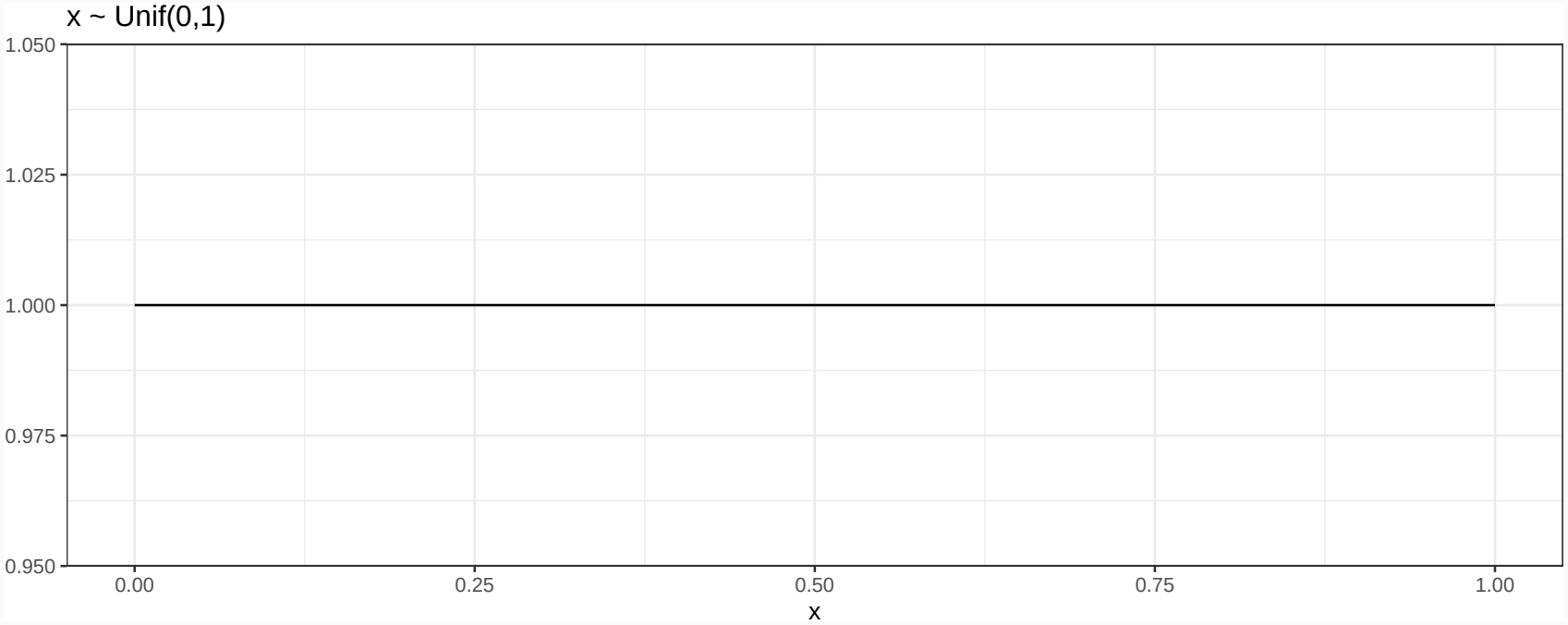
We can express this lack of insight by stating our prior beliefs:

Prior for p

$$p \sim \text{Uniform}(0, 1)$$

Probability densities

Our prior, a uniform distribution for a proportion on [0,1]



So far

We are assuming that the data have a Bernoulli likelihood. We have no idea what p is.

So far

We are assuming that the data have a Bernoulli likelihood. We have no idea what p is.

Let's collect data!

```
coins<-rbinom(15, 1, prob = 0.4)
coins
```

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

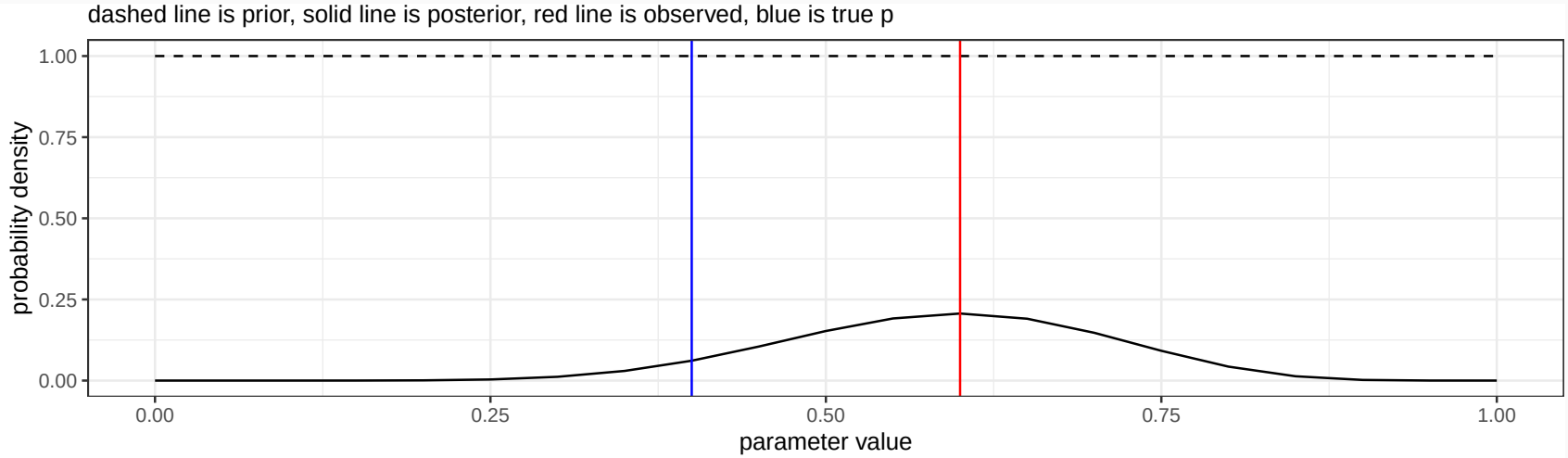
Now let's compute a likelihood for possible values of p

```r
p<-data.frame(p_seq = seq(0, 1, 0.05))

### use dbinom to compute the probabilities
p<-p %>%
  mutate(likelihood = dbinom(sum(coins), 15, p_seq),
         prior = 1,
         posterior = prior * likelihood)

p
```

```
##    p_seq   likelihood prior    posterior
## 1   0.00 0.000000e+00     1 0.000000e+00
## 2   0.05 7.185810e-09     1 7.185810e-09
## 3   0.10 2.659862e-06     1 2.659862e-06
## 4   0.15 7.256697e-05     1 7.256697e-05
## 5   0.20 6.717597e-04     1 6.717597e-04
## 6   0.25 3.398065e-03     1 3.398065e-03
## 7   0.30 1.159000e-02     1 1.159000e-02
## 8   0.35 2.975066e-02     1 2.975066e-02
## 9   0.40 6.121411e-02     1 6.121411e-02
## 10  0.45 1.048318e-01     1 1.048318e-01
## 11  0.50 1.527405e-01     1 1.527405e-01
## 12  0.55 1.914006e-01     1 1.914006e-01
## 13  0.60 2.065976e-01     1 2.065976e-01
## 14  0.65 1.905604e-01     1 1.905604e-01
## 15  0.70 1.472360e-01     1 1.472360e-01
## 16  0.75 9.174777e-02     1 9.174777e-02
## 17  0.80 4.299262e-02     1 4.299262e-02
## 18  0.85 1.320450e-02     1 1.320450e-02
```

And visualize it!

```
ggplot(p,
       aes(x = p_seq, y = prior)) +
  geom_line(lty = 2) +
  geom_line(aes(y = posterior)) +
  geom_vline(xintercept = sum(coins)/15, color = "red") +
  geom_vline(xintercept = 0.4, color = "blue") +
  labs(subtitle = "dashed line is prior, solid line is posterior, r
       y = "probability density", x= "parameter value")
```



What if we used a different prior?

What if we brought in our prior knowledge that $p(coin) = 0.5$.

```
p<-data.frame(p_seq = seq(0, 1, 0.05))

### use dbinom to compute the probabilities
p<-p %>%
  mutate(likelihood = dbinom(sum(coins), 15, p_seq),
         prior = ifelse(p_seq==0.5, 1, 0),
         posterior = prior * likelihood)

### weird...
p
```

```
##    p_seq   likelihood prior posterior
## 1   0.00 0.000000e+00     0 0.0000000
## 2   0.05 7.185810e-09     0 0.0000000
## 3   0.10 2.659862e-06     0 0.0000000
## 4   0.15 7.256697e-05     0 0.0000000
## 5   0.20 6.717597e-04     0 0.0000000
## 6   0.25 3.398065e-03     0 0.0000000
## 7   0.30 1.159000e-02     0 0.0000000
## 8   0.35 2.975066e-02     0 0.0000000
## 9   0.40 6.121411e-02     0 0.0000000
## 10  0.45 1.048318e-01     0 0.0000000
## 11  0.50 1.527405e-01     1 0.1527405
## 12  0.55 1.914006e-01     0 0.0000000
## 13  0.60 2.065976e-01     0 0.0000000
## 14  0.65 1.905604e-01     0 0.0000000
## 15  0.70 1.473269e-01     0 0.0000000
```

Back to regression

The mpg data again

```
m1_bayes<-stan_glm(mpg ~ wt,
                   data = mtcars)

### check out the default weakly informative priors
prior_summary(m1_bayes)
```

```
## Priors for model 'm1_bayes'
## ------
## Intercept (after predictors centered)
##   Specified prior:
##     ~ normal(location = 20, scale = 2.5)
##   Adjusted prior:
##     ~ normal(location = 20, scale = 15)
##
## Coefficients
##   Specified prior:
##     ~ normal(location = 0, scale = 2.5)
##   Adjusted prior:
##     ~ normal(location = 0, scale = 15)
##
## Auxiliary (sigma)
##   Specified prior:
##     ~ exponential(rate = 1)
##   Adjusted prior:
##     ~ exponential(rate = 0.17)
## ------
## See help('prior_summary.stanreg') for more details
```

What do those priors tell us?

$$p(\beta_0) \sim N(20, 15)$$

$$p(\beta_1) \sim N(0, 15)$$

Interpreting in a Bayesian vs frequentist context

```
confint(m1)
```

```
##                 2.5 %    97.5 %
## (Intercept) 33.450500 41.119753
## wt          -6.486308 -4.202635
```

```
posterior_interval(m1_bayes)
```

```
##                      5%       95%
## (Intercept) 33.954226 40.402936
## wt          -6.292020 -4.338039
## sigma        2.543167  3.931898
```

Working with Bayesian models: posterior parameter inference

```
## let's play with this
m1_bayes_post<-as_tibble(m1_bayes)
glimpse(m1_bayes_post)
```

```
## Rows: 4,000
## Columns: 3
## $ `(Intercept)` <dbl> 40.55998, 37.10447, 35.78866, 36.32431, 37.76495, 36....
## $ wt            <dbl> -6.348470, -5.361067, -5.018177, -5.090330, -5.395205...
## $ sigma         <dbl> 2.807843, 4.369583, 2.338327, 3.011319, 3.102185, 3.0...
```

Posterior prediction

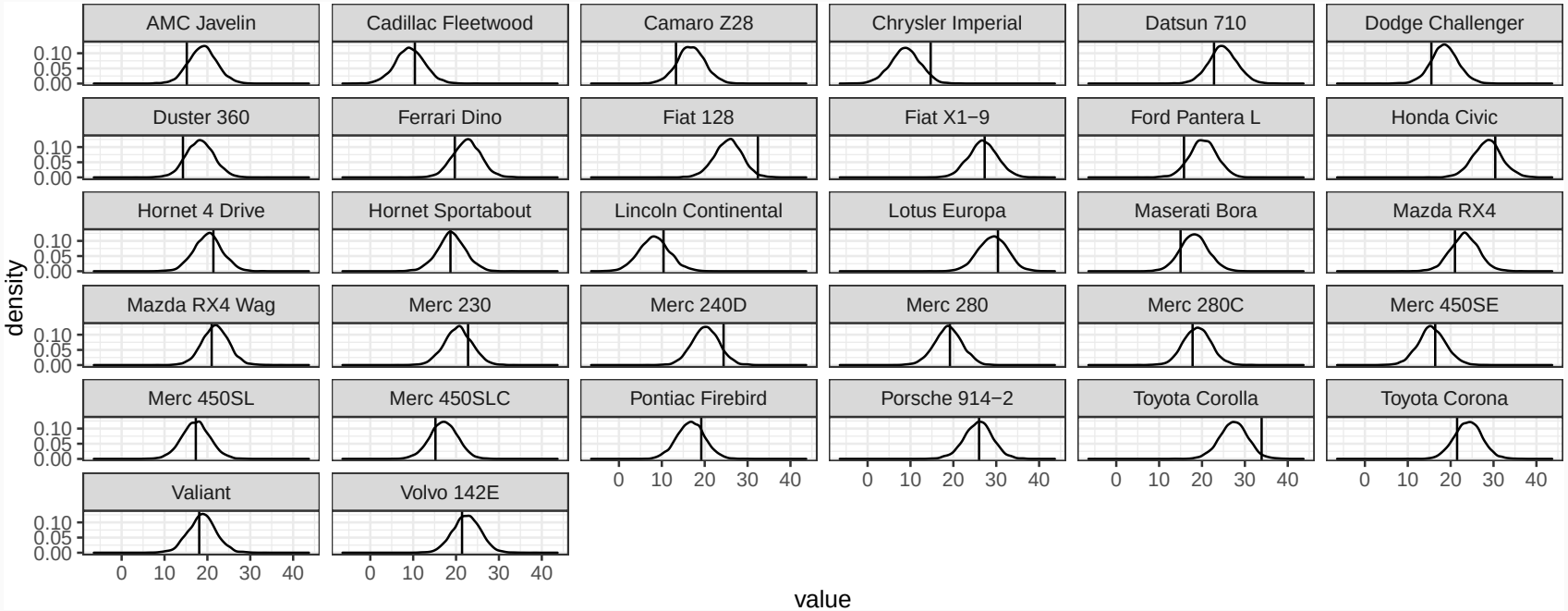
```
### predict over each observed value of mpg
### and make it tidy, because tidy is nice
m1_preds<-as_tibble(posterior_predict(m1_bayes)) %>%
  pivot_longer(cols = everything())

head(m1_preds)
```

```
## # A tibble: 6 x 2
##   name              value
##   <chr>             <ppd>
## 1 Mazda RX4         19.78678
## 2 Mazda RX4 Wag     21.49429
## 3 Datsun 710        23.86908
## 4 Hornet 4 Drive    16.38634
## 5 Hornet Sportabout 18.98480
## 6 Valiant           20.97747
```

```
## add the observed mpg value back in
m1_preds<-m1_preds %>%
  left_join(mtcars %>%
  mutate(name = rownames(mtcars)) %>%
  select(name, mpg))
```

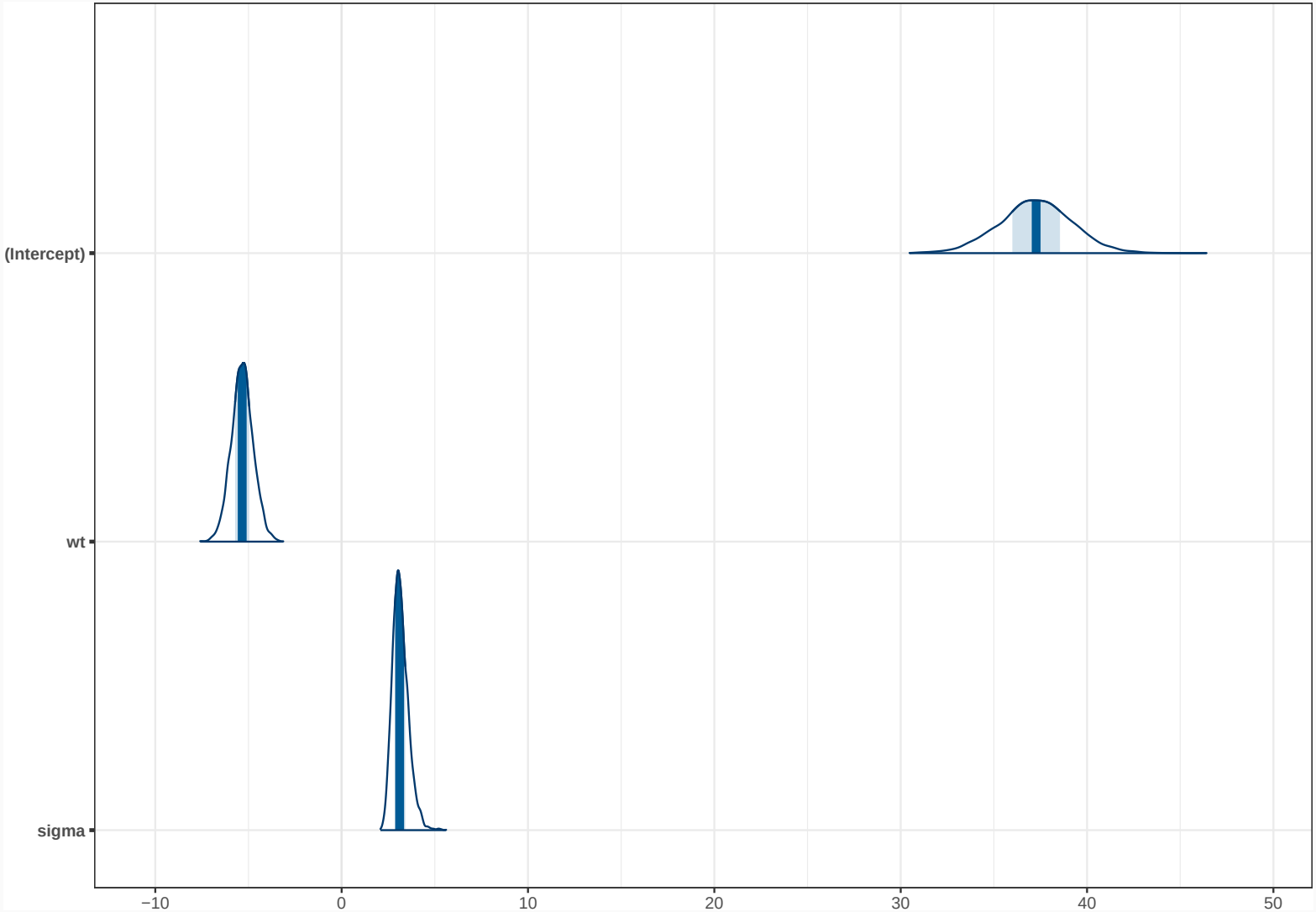
And then plot it!

```
ggplot(m1_preds,
       aes(x = value)) +
  geom_density() +
  geom_vline(aes(xintercept = mpg)) +
  facet_wrap(~name)
```



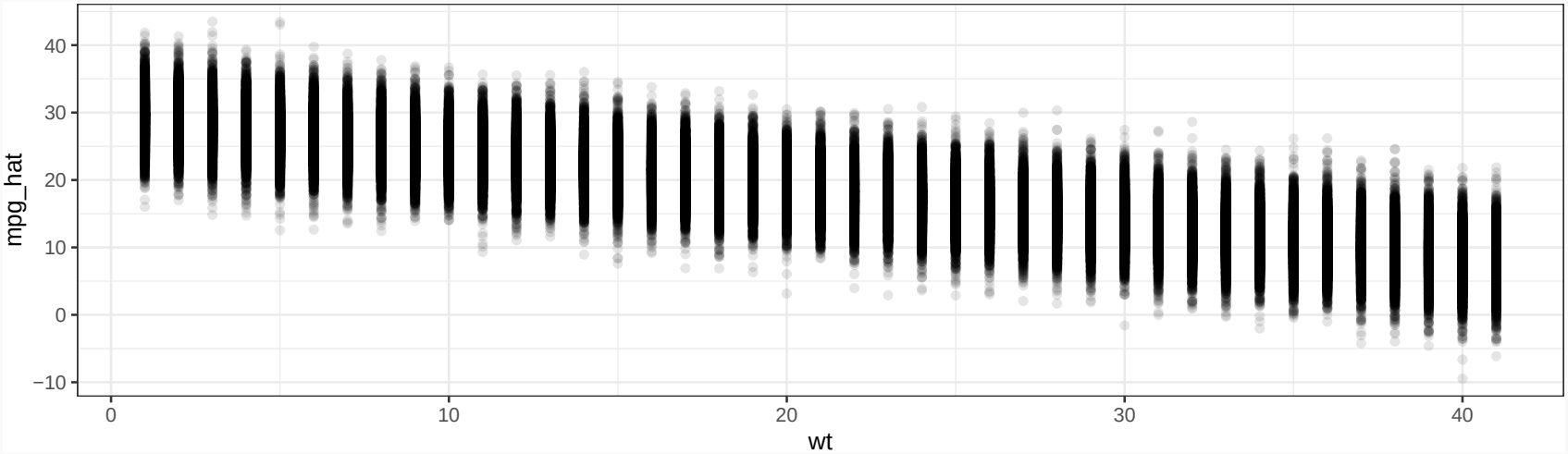
Well, this is easy

```
plot(m1_bayes, "areas")
```



Let’s make a more informative posterior prediction

```
fake_data<-tibble(wt = seq(1.5, 5.5, 0.1))

pred_out<-as_tibble(posterior_predict(m1_bayes, newdata = fake_data)) %>%
  pivot_longer(cols = everything(), names_to = "wt", values_to = "mpg_hat") %>%
  mutate(wt = as.numeric(wt), mpg_hat = as.numeric(mpg_hat))

ggplot(pred_out, aes(x = wt, y = mpg_hat)) +
  geom_point(alpha = 0.1)
```



More exciting models

Maybe fuel efficiency is a joint function of weight and engine power?

```
m2_bayes<-stan_glm(mpg ~ scale(wt) + scale(hp),
                   data = mtcars)
posterior_interval(m2_bayes)
```

```
##                    5%       95%
## (Intercept) 19.311052 20.884311
## scale(wt)   -4.822757 -2.680720
## scale(hp)   -3.266737 -1.124715
## sigma        2.168289  3.334813
```

What's this scale() nonsense?

Interactions!

```
m3_bayes<-stan_glm(mpg ~ scale(wt) + scale(hp) +
                     scale(wt)*scale(hp),
                   data = mtcars)

posterior_interval(m3_bayes)
```

```
##                             5%        95%
## (Intercept)          18.0542476 19.764309
## scale(wt)            -4.9081876 -3.152201
## scale(hp)            -2.9756270 -1.201416
## scale(wt):scale(hp)   0.9746008  2.732735
## sigma                 1.7908083  2.840580
```

Which one is a better fit?

We can use leave-one-out cross validation to compare model fits

```
loo_m1<-loo(m1_bayes)
loo_m2<-loo(m2_bayes)
loo_m3<-loo(m3_bayes)
loo_compare(loo_m1, loo_m2, loo_m3)
```

```
##           elpd_diff se_diff
## m3_bayes    0.0       0.0
## m2_bayes   -5.8       3.0
## m1_bayes  -10.3       2.8
```

Homework

Repeat HW3 with Bayesian regression models.

Use appropriate techniques to describe your findings.