

Example R notebook

Loading packages

The very first thing that you would like to do is loading the required R libraries. Lets check whether you have successfully installed all the packages.

```
library(tidyverse)
library(emmeans)
library(brms)
library(tidybayes)
library(bayestestR)
```

You will probably see some start message, and several warnings. If you don't see any errors, then with 99% you are probably fine with this step.

Obtaining the dataset.

Most of the datasets will be available through the course online repository (or some other repositories). Lets try to get the example dataset.

```
library(foreign)
crime_data <- read.dta("https://stats.idre.ucla.edu/stat/data/crime.dta")
```

Initial lookup at the data

There are several ways to look at the dataset. Lets try a simple `glimpse`.

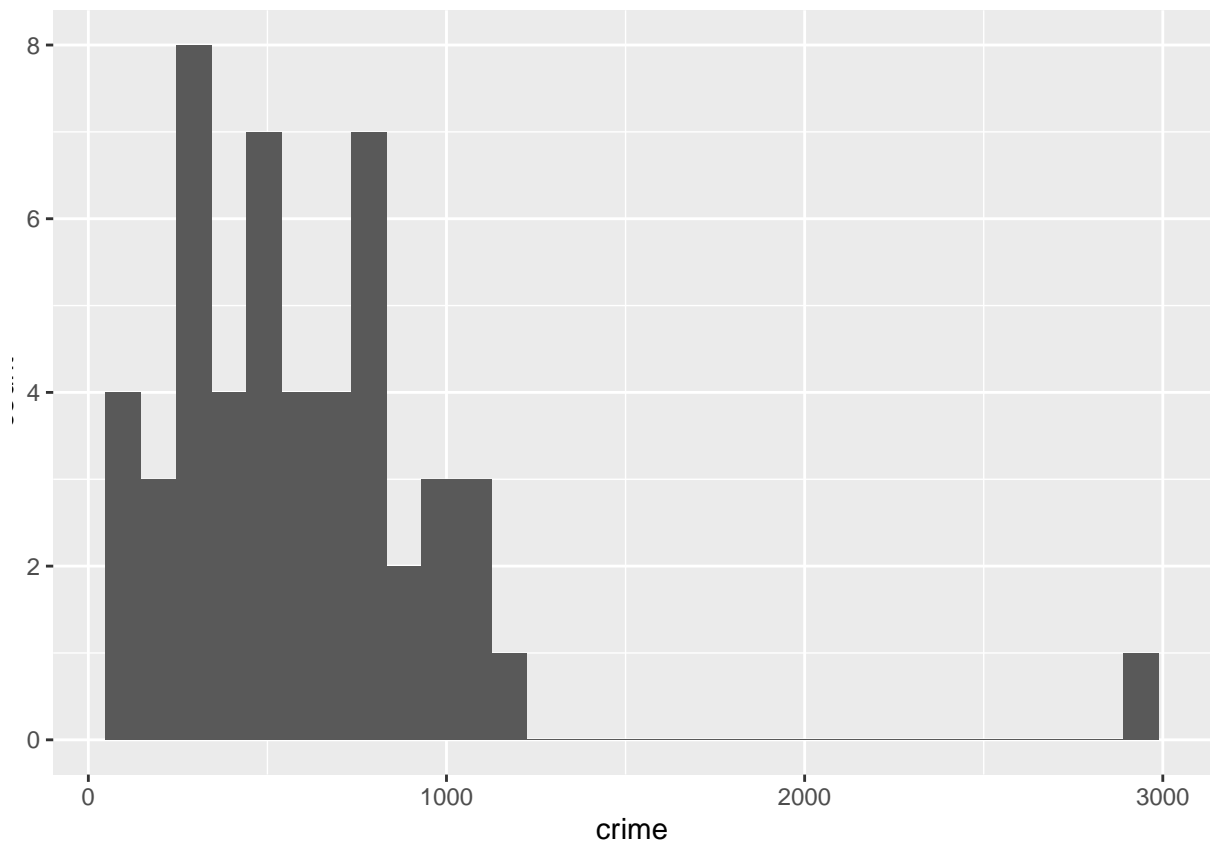
```
crime_data %>%
  glimpse()

## Observations: 51
## Variables: 9
## $ sid      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16...
## $ state    <chr> "ak", "al", "ar", "az", "ca", "co", "ct", "de", "fl",...
## $ crime    <int> 761, 780, 593, 715, 1078, 567, 456, 686, 1206, 723, 2...
## $ murder   <dbl> 9.0, 11.6, 10.2, 8.6, 13.1, 5.8, 6.3, 5.0, 8.9, 11.4,...
## $ pctmetro <dbl> 41.8, 67.4, 44.7, 84.7, 96.7, 81.8, 95.7, 82.7, 93.0,...
## $ pctwhite <dbl> 75.2, 73.5, 82.9, 88.6, 79.3, 92.5, 89.0, 79.4, 83.5,...
## $ pcths    <dbl> 86.6, 66.9, 66.3, 78.7, 76.2, 84.4, 79.2, 77.5, 74.4,...
## $ poverty  <dbl> 9.1, 17.4, 20.0, 15.4, 18.2, 9.9, 8.5, 10.2, 17.8, 13...
## $ single   <dbl> 14.3, 11.5, 10.7, 12.1, 12.5, 12.1, 10.1, 11.4, 10.6,...
```

Plotting some of the distributions.

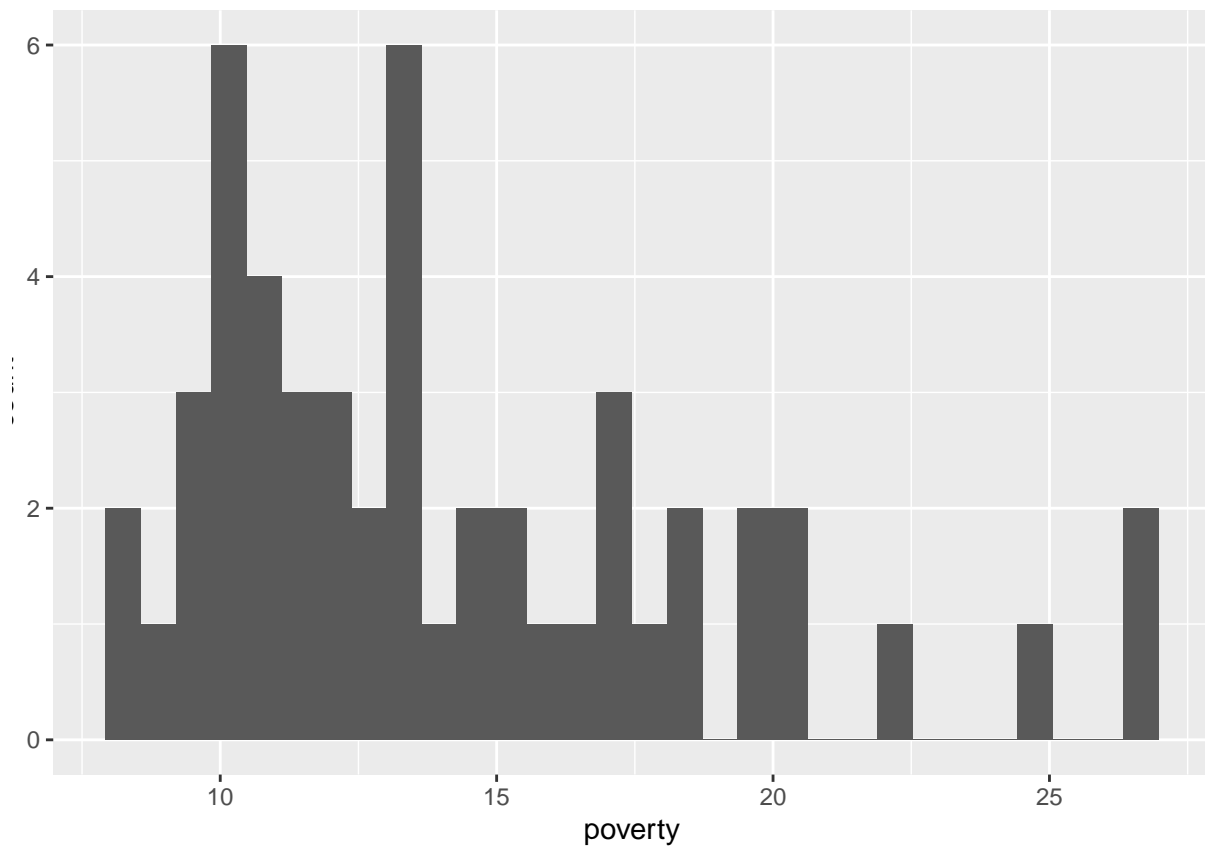
We are interested in the crime level through states. Lets plot a simple histogram.

```
crime_data %>%
  ggplot(aes(crime)) +
  geom_histogram()
```



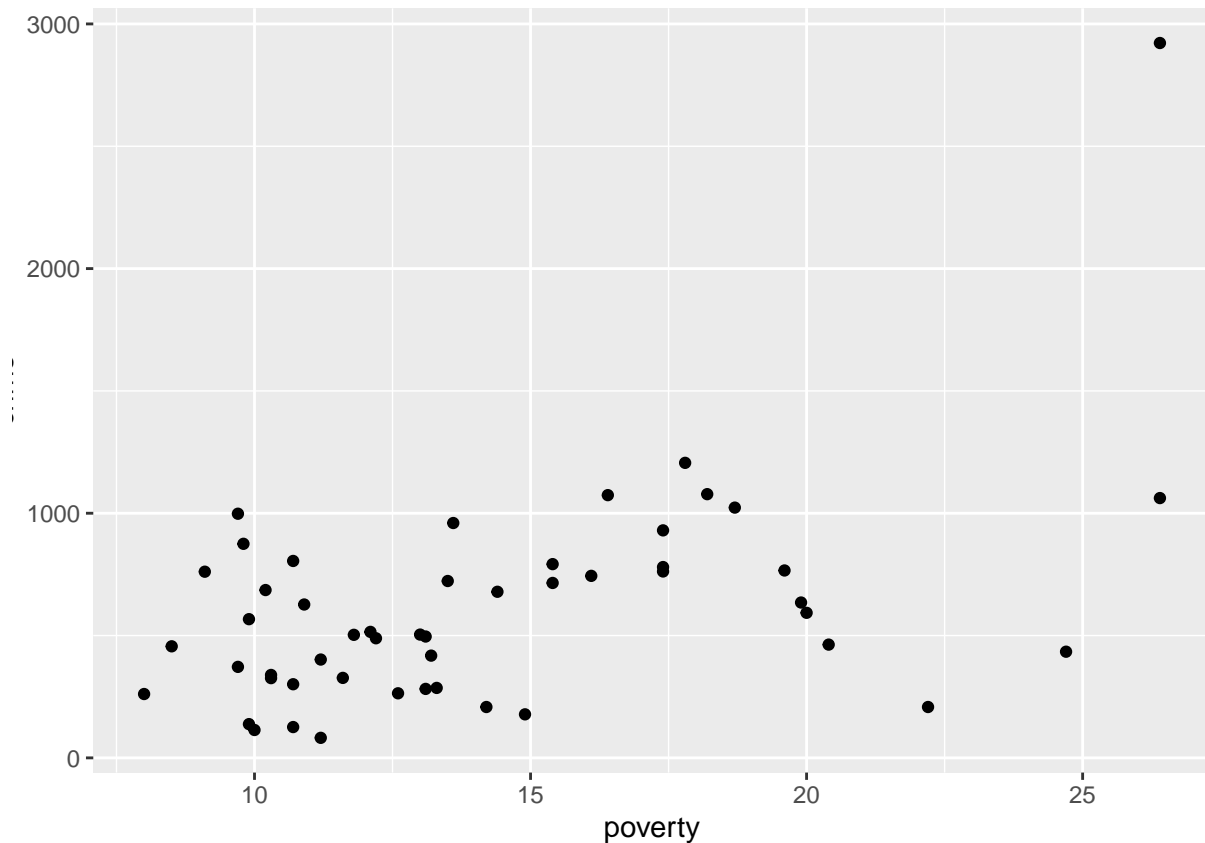
The distribution looks fine apart from a single outlier. Lets plot one the predictors of crime - poverty.

```
crime_data %>%  
  ggplot(aes(poverty)) +  
  geom_histogram()
```



Now lets plot crime against poverty.

```
crime_data %>%  
  ggplot(aes(poverty, crime)) +  
  geom_point()
```



Creating a simple Bayesian regression model.

Lets fit crime against poverty. This is your Hello world! to Bayesian modelling. Don't worry if you don't understand what you are doing or what is happening. This just a quick intro to get you the feeling of how we will work through the course.

```
fit <- brm(crime ~ poverty,
  data = crime_data,
  prior = prior(normal(0, 10), class = b))
```

```
##
## SAMPLING FOR MODEL '4666657389157456f312417abee138c4' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%] (Warmup)
## Chain 1: Iteration:   200 / 2000 [ 10%] (Warmup)
## Chain 1: Iteration:   400 / 2000 [ 20%] (Warmup)
## Chain 1: Iteration:   600 / 2000 [ 30%] (Warmup)
## Chain 1: Iteration:   800 / 2000 [ 40%] (Warmup)
## Chain 1: Iteration:  1000 / 2000 [ 50%] (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%] (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%] (Sampling)
```

```

## Chain 1: Iteration: 1400 / 2000 [ 70%] (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%] (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%] (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%] (Sampling)
## Chain 1:
## Chain 1: Elapsed Time: 0.055393 seconds (Warm-up)
## Chain 1: 0.010983 seconds (Sampling)
## Chain 1: 0.066376 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '4666657389157456f312417abee138c4' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 4e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration: 1 / 2000 [ 0%] (Warmup)
## Chain 2: Iteration: 200 / 2000 [ 10%] (Warmup)
## Chain 2: Iteration: 400 / 2000 [ 20%] (Warmup)
## Chain 2: Iteration: 600 / 2000 [ 30%] (Warmup)
## Chain 2: Iteration: 800 / 2000 [ 40%] (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%] (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%] (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%] (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%] (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%] (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%] (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%] (Sampling)
## Chain 2:
## Chain 2: Elapsed Time: 0.050445 seconds (Warm-up)
## Chain 2: 0.0118 seconds (Sampling)
## Chain 2: 0.062245 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '4666657389157456f312417abee138c4' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 6e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration: 1 / 2000 [ 0%] (Warmup)
## Chain 3: Iteration: 200 / 2000 [ 10%] (Warmup)
## Chain 3: Iteration: 400 / 2000 [ 20%] (Warmup)
## Chain 3: Iteration: 600 / 2000 [ 30%] (Warmup)
## Chain 3: Iteration: 800 / 2000 [ 40%] (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%] (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%] (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%] (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%] (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%] (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%] (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%] (Sampling)

```

```

## Chain 3:
## Chain 3: Elapsed Time: 0.062326 seconds (Warm-up)
## Chain 3:           0.011165 seconds (Sampling)
## Chain 3:           0.073491 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '4666657389157456f312417abee138c4' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 4e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [ 0%] (Warmup)
## Chain 4: Iteration:   200 / 2000 [ 10%] (Warmup)
## Chain 4: Iteration:   400 / 2000 [ 20%] (Warmup)
## Chain 4: Iteration:   600 / 2000 [ 30%] (Warmup)
## Chain 4: Iteration:   800 / 2000 [ 40%] (Warmup)
## Chain 4: Iteration:  1000 / 2000 [ 50%] (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%] (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%] (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%] (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%] (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%] (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%] (Sampling)
## Chain 4:
## Chain 4: Elapsed Time: 0.053084 seconds (Warm-up)
## Chain 4:           0.01137 seconds (Sampling)
## Chain 4:           0.064454 seconds (Total)
## Chain 4:

```

Summarising our results

Lets summarise our results.

```
fit
```

```

## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: crime ~ poverty
## Data: crime_data (Number of observations: 51)
## 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    336.45    129.34    85.30   595.36 1.00     3366     3124
## poverty       19.03      8.14      2.58    34.72 1.00     3445     3462
##
## Family Specific Parameters:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma    410.59     43.12    335.56   503.23 1.00     3358     2845
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample

```

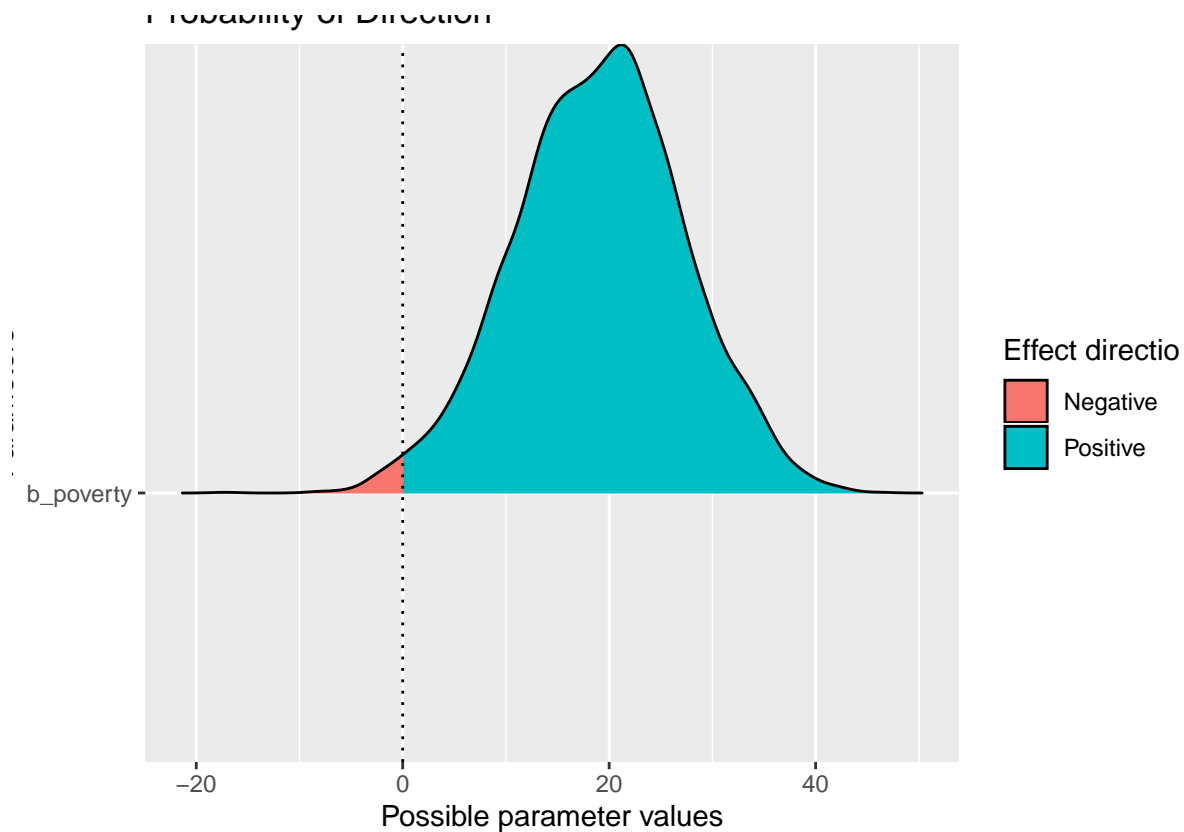
```
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

It seems like poverty is positively and reliably associated with crime level (at the state level).

Plotting our posterior.

Lets see to what extent the obtained posterior distribution supports positive association.

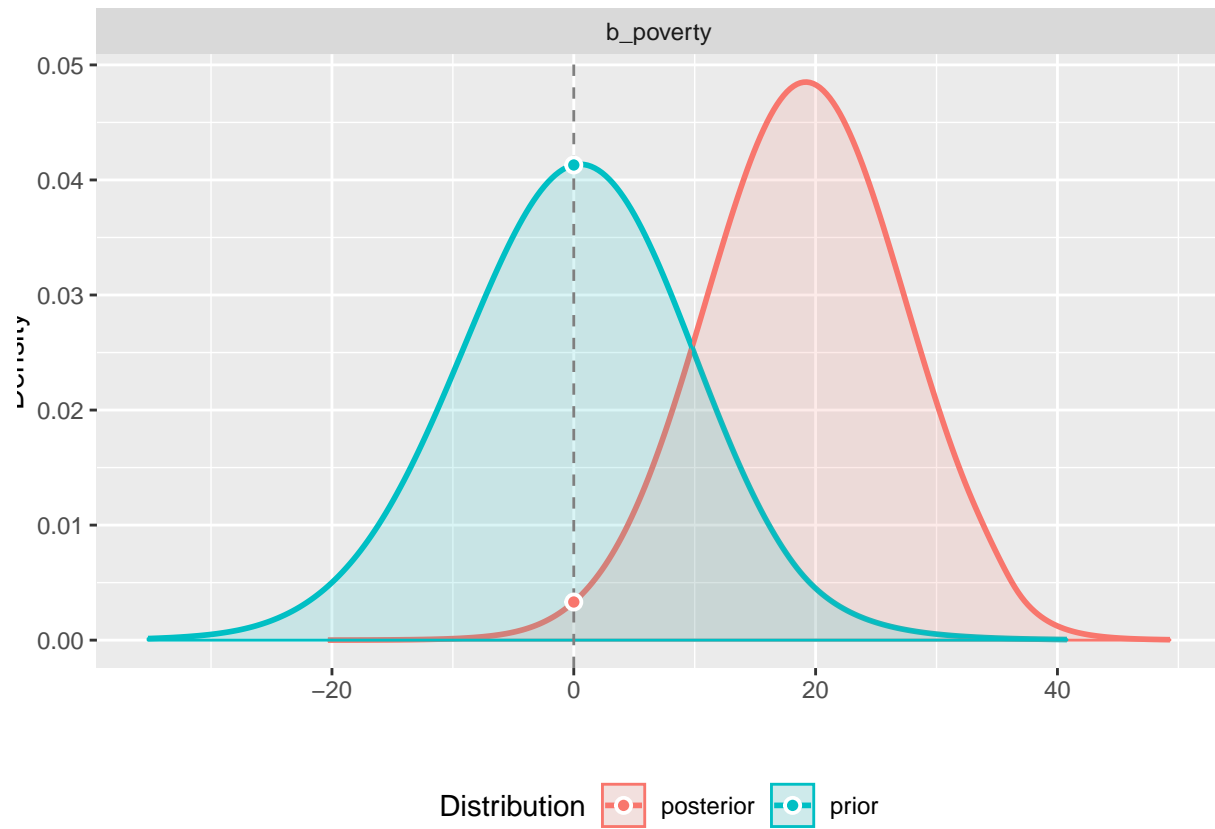
```
pd_fit <- p_direction(fit)
plot(pd_fit)
```



Plotting posterior against prior

Now lets compare posterior distribution to prior distribution.

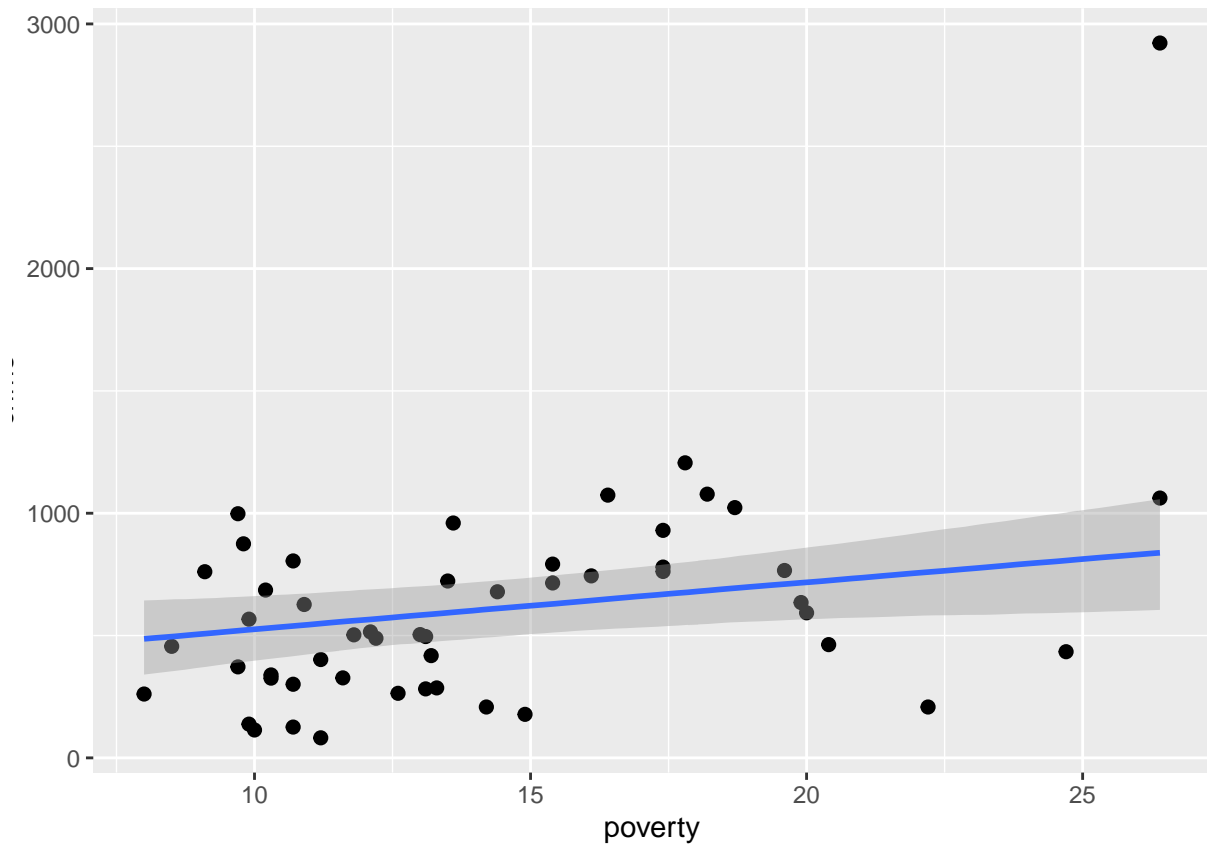
```
bf_fit <- bayesfactor_parameters(fit)
plot(bf_fit)
```



Plotting predictions against data

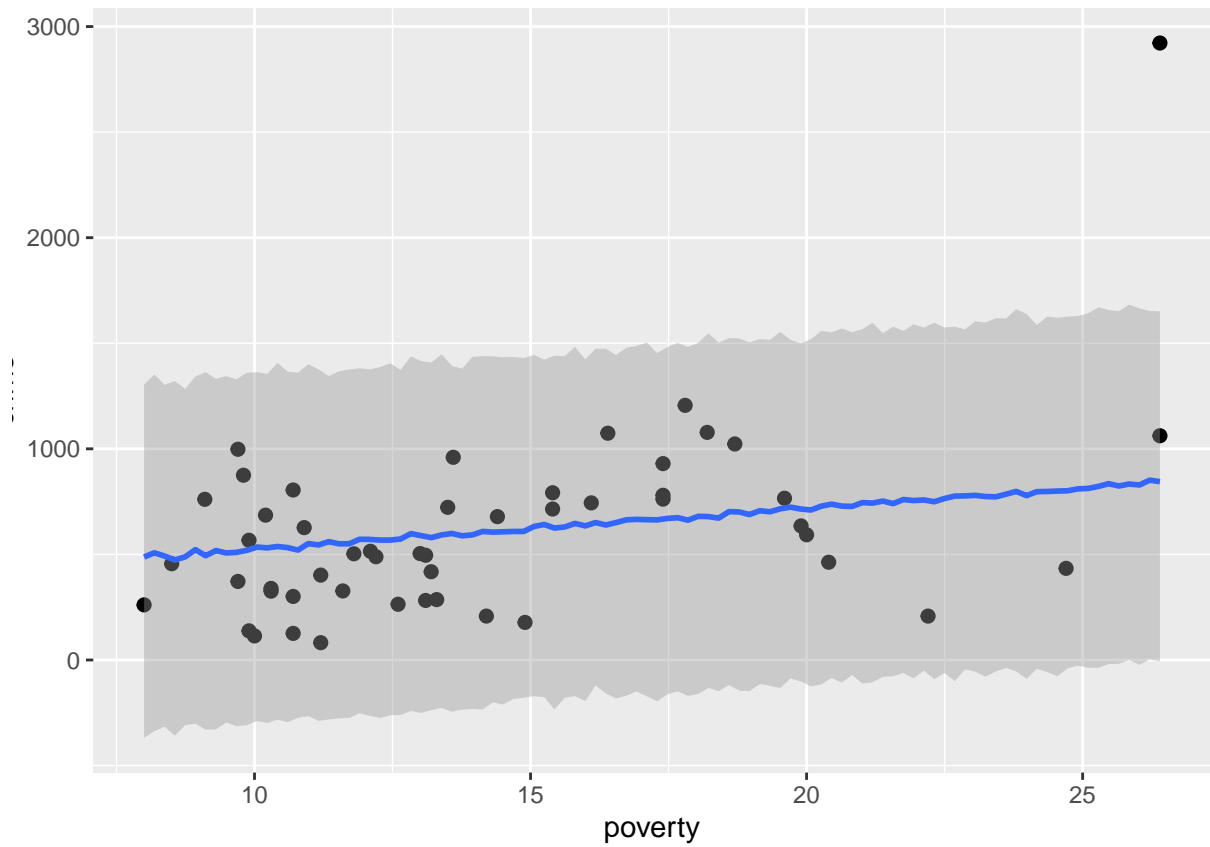
Now lets plot the fitted regression line against the collected data.

```
marginal_effects(fit, effects = "poverty") %>%  
  plot(points = T)
```

If you would like to fit predicted crime level given the poverty level it is also quite simple.

```
marginal_effects(fit, effects = "poverty", method = "predict") %>%  
  plot(points = T)
```



If you find the code in this notebook hard to follow, DON'T PANIC. It will get easier with time. I would also recommend you to go through some online tutorials. For example Hadley Wickham's R for Data Science.