An introduction to Stan for applied Bayesian inference

FW 891

Christopher Cahill

6 September 2023

Quantitative Fisheries Center
MICHIGAN STATE UNIVERSITY

Purpose

- Learn the basic syntax of the Stan language
- Write code to elicit simple models and implement Bayesian inference in Stan
- Use the cmdstanr interface
- Develop familiarity with a few packages that make your life easier
- Walk through some model diagnostics
- Make sure you have these programs/packages installed

Installing CmdStanR

- See the installation instructions here

```r
library(cmdstanr)
# use a built in file that comes with cmdstanr:
file <- file.path(
  cmdstan_path(), "examples",
  "bernoulli", "bernoulli.stan"
)
mod <- cmdstan_model(file)
```

see also CmdStan user's guide

Now let’s make sure it works

```
# tagged list where names correspond to the .stan data block
stan_data <- list(N = 10, y = c(0, 1, 0, 0, 0, 0, 0, 0, 0, 1))

fit <- mod$sample(
  data = stan_data,
  seed = 123,
  chains = 4,
  parallel_chains = 4,
  refresh = 500 # print update every 500 iters
)
```

Do the bottom numbers match up?

```
Running MCMC with 4 parallel chains...

Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1 finished in 0.0 seconds.
Chain 2 finished in 0.0 seconds.
Chain 3 finished in 0.0 seconds.
Chain 4 finished in 0.0 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.0 seconds.
Total execution time: 0.4 seconds.
```

```
fit$summary() # you should get these numbers:
```

```
# A tibble: 2 × 10
  variable   mean median    sd   mad      q5    q95  rhat ess_bulk ess_tail
  <chr>     <num>  <num> <num> <num>   <num>  <num> <num>    <num>    <num>
1 lp__     -7.26  -6.99  0.719 0.329 -8.73   -6.75   1.00    1658.    1861.
2 theta     0.246  0.231 0.118 0.118  0.0811  0.463  1.00    1378.    1236.
```

Presumably this broke someone

image credit

Onward!

Stan: the basics

Stan is a probablistic modeling language https://mc-stan.org/

- Freely available
- Implements HMC, and an algorithm called NUTS
  - No U-Turn Sampler
  - We are using it for full Bayesian inference, but it can do other things too (we will not talk about these things)
- The Stan documentation and community is legendary in my opinion, albeit dense at times

Using Stan requires writing a `.stan` file

- Coding in Stan is something of a cross between R, WINBUGS/JAGS, and C++
- It is a Turing complete programming language
- Stan requires you to be explicit
    - Need to tell it whether something is a real, integer, vector, matrix, array, etc.
    - Lines need to end in a `;`
- A `.stan` file relies on program blocks to read in your data and contruct your model
- Many built in functions you can use
- Why must we confront misery of a new language?

A linear regression in Stan

Let's build a linear regression model, which can be written a few ways:

$$y_i = \alpha + \beta x_i + \epsilon_i \quad \text{where} \quad \epsilon_i \sim \text{normal}(0, \sigma).$$

which is the same as

$$y_i - (\alpha + \beta X_i) \sim \text{normal}(0, \sigma)$$

and reducing further:

$$y_i \sim \text{normal}(\alpha + \beta X_i, \sigma).$$

Linear regression in Stan cont'd

- Let's build a simple linear regression model in Stan

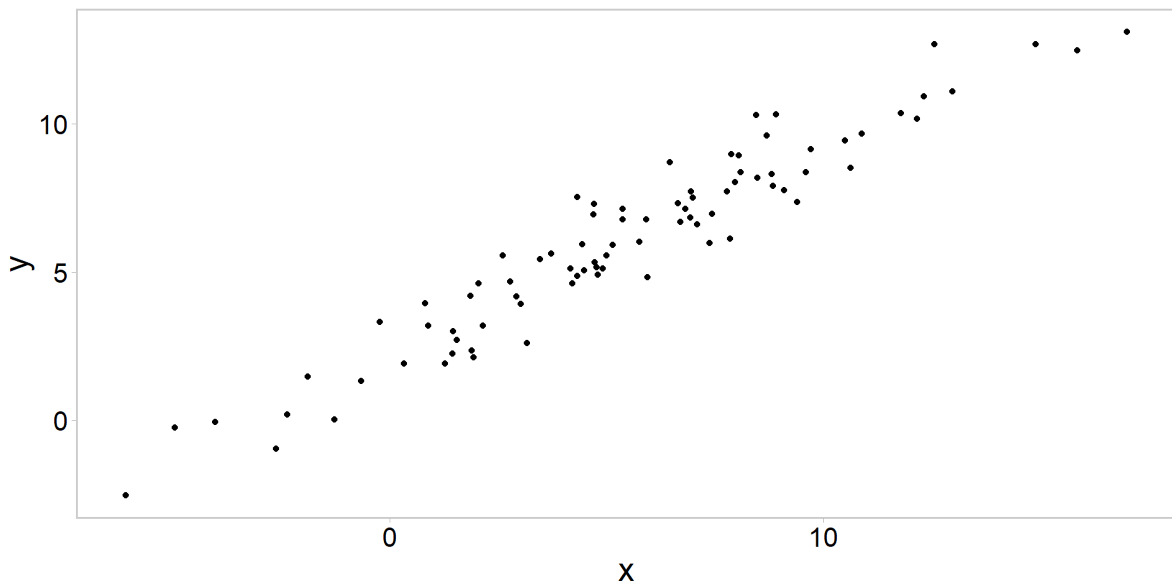
The data

- What do we do when we get some data?

Always plot the data

```r
library(tidyverse)
library(ggqfc)
library(cmdstanr)

data <- readRDS("data/linreg.rds")
p <- data %>% ggplot(aes(y = y, x = x)) +
geom_point() + theme_qfc() +
  theme(text = element_text(size = 20))
p
```



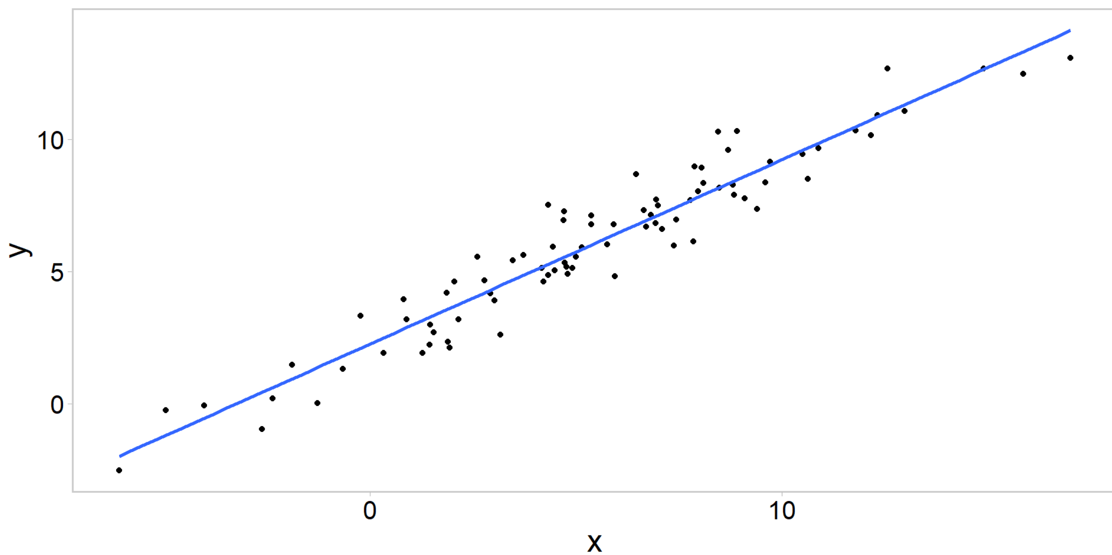
Always plot the data

```r
p + geom_smooth(method = lm, se = F)
```

```r
lm(data$y ~ data$x) # fit y = a + bx + e, where e ~ N(0, sd)
```

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

Coefficients:
(Intercept)       data$x
     2.2559       0.6979
```



Thinking through our model

$$y_i \sim \text{normal}(\alpha + \beta X_i, \sigma).$$

signal = deterministic component + random component

Thinking through our model

$$y_i \sim \text{normal}(\alpha + \beta X_i, \sigma).$$

signal = deterministic component + random component

If $\mu_i \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$, then for $y_i \in \mathbb{R}$,

$$\text{Normal}(y_i \mid \mu_i, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \left(\frac{y_i - \mu_i}{\sigma} \right)^2 \right)$$

Thinking through our model

$$y_i \sim \text{normal}(\alpha + \beta X_i, \sigma).$$

signal = deterministic component + random component

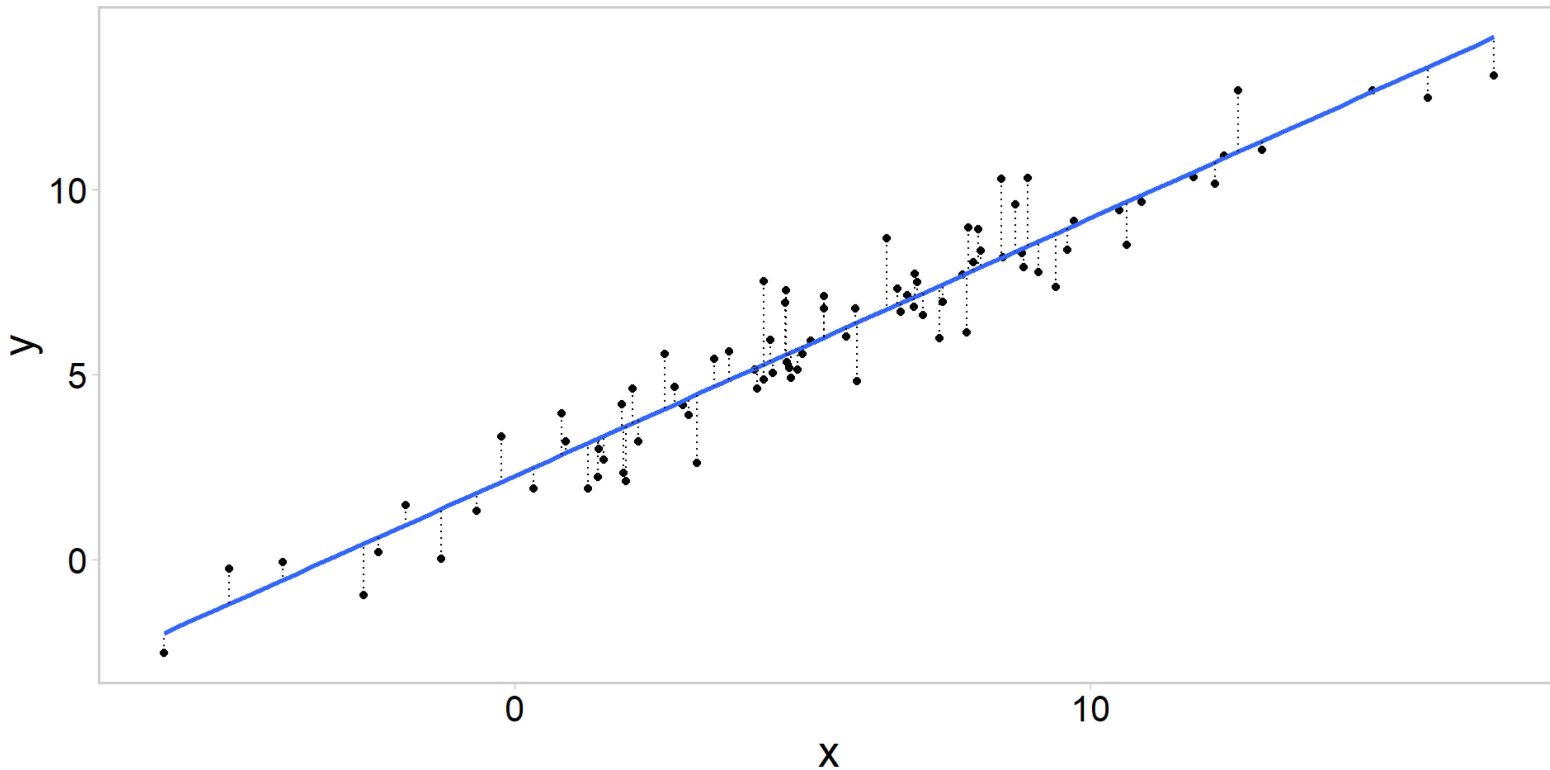
If $\mu_i \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$, then for $y_i \in \mathbb{R}$,

$$\text{Normal}(y_i \mid \mu_i, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{y_i - \mu_i}{\sigma}\right)^2\right)$$

where, $\mu_i = \alpha + \beta X_i$

Thinking through our model

$$y_i \sim \text{normal}(\alpha + \beta X_i, \sigma).$$



Writing our first `.stan` model

Code to do what we are going through is in the `week2/` Github directory

`linreg.R` and `linreg.stan`

mc-stan.org

Structure of a `.stan` file

```
// this is a comment
// program block demonstration
data{
  // read in data here -- this section is executed one time per Stan run
}
transformed data {
  // transform the data here -- this section is also executed one time per Stan run
}
parameters {
  // declare the **estimated** parameters here
}
transformed parameters{
  // this section takes parameter estimates and data (or transformed data)
  // and transforms them for use later on in model section
}
model{
  // this section specifies the prior(s) and likelihood terms,
  // and defines a log probability function (i.e., log posterior) of the model
}
generated quantities{
  // this section creates derived quantities based on parameters,
  // models, data, and (optionally) pseudo-random numbers.
}
```

- Can also write custom functions (although we won't in this class)

In words, rather than code

As per the comments in the code, each of the program blocks does certain stuff

- `data{ }` reads data into the .stan program
- `transformed data{ }` runs calculations on those data (once)
- `parameters{ }` declares the ***estimated*** parameters in a Stan program
- `transformed parameters{ }` takes the parameters, data, and transformed data, and calculates stuff you need for your model
- `model{ }` constructs a log probability function:
  - $log(posterior) = log(priors) + log(likelihood)$
- `generated quantities{ }` is only executed after you have your sampled posterior
  - useful for calculating derived quantities given your model, data, and parameters

Priors in Stan

- If you don't specify priors, *Stan will specify flat priors for you*
  - Not always a good thing, and it can lead to problems
- In this class we are either going to use vague or uninformative priors
  - Or, we will use informative priors that incorporate domain expertise or information from previous studies
- When we say this prior is "weakly informative," what we mean is that if there's a large amount of data, the likelihood will dominate, and the prior will not be important
  - Prior can often only be understood in the context of the likelihood (Gelman et al. 2017; see also prior recommendations in Stan)

see arguments in Kery and Schaub 2012; Gelman et al. 2017; McElreath 2023

Standardizing covariates

$$z_i = \frac{x_i - \mu}{sd_X}$$

Stan reference

https://academic.oup.com/jrsssa/article/182/2/389/7070184?login=false