ISYE 6420 - BUGS to Stan

Josh Blakely

2/15/23

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

```
1 + 1
```

```
[1] 2
```

install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))

Part I

UNIT 1

About this course and website

The course was originally developed at Georgia Tech in 2004 by Professor Brani Vidakovic, who is now the head of the Texas A&M Department of Statistics. Many of the individual examples are related to biostatistics (Prof. Vidakovic wrote the textbook *Engineering Biostatistics*), but the methods are broadly applicable.

> See also Professor Vidakovic's publication history.
>
> Professor Joseph's.

This site mostly follows the original course outline. Each example lists the corresponding lecture video and contains a download link to the original code file(s). Only the lecture video where the professor goes over the example code will be listed, but there may be other relevant lectures that you'll need to watch. Any necessary data will either have a download link or, if the data is compact enough, will be included in the code.

This is a supplement to the Canvas site, not a replacement!

Other recommended resources

These are not required for the class, but they might be helpful. Prof. Vidakovic's lectures often assume that the student has a certain amount of background knowledge, so if you feel lost or if you just want to dive deeper into the subject check them out.

Textbooks and courses

Statistical Rethinking by Richard McElreath is a great book for gaining intuition about Bayesian inference and modeling in general.

- main site
- lecture videos
- R and Stan code examples.

Bayesian Data Analysis by Gelman, Carlin, Stern, Dunson, Vehtari, and Rubin goes into more mathematical theory than *Statistical Rethinking*. I use it as a reference - not planning to try reading this one all the way through!

- main site
- pdf

Blogs

- Andrew Gelman, author of *Bayesian Data Analysis* (above), has a blog: Statistical Modeling, Causal Inference, and Social Science.
- Dan Simpson, one of the maintainers of the Stan PPL, has a blog called Un garçon pas comme les autres (Bayes) with opinionated and funny deep dives into various Bayesian topics. Warning: NSFW language.
- Count Bayesie: Probably a probability blog by Will Kurt.
- Michael Betancourt, another Stan developer, has a series of incredibly in-depth posts and notebooks on Bayesian modeling.
- PyMC developer Austin Rochford's blog has a lot of good posts.
- Another PyMC developer, Oriol Abril, posts some really helpful PyMC examples.

Podcasts

- Learn Bayes Stats by Alex Andorra, one of the PyMC developers.

Other

- Stan User's Guide
- Aaron Reding's PyMC

Part II

UNIT 2

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

```
install.packages("rstan", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))
install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos"))
```

Part III

UNIT 3

2 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

```
install.packages("rstan", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))
install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos"))
```

Part IV

UNIT 4

3 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

```
install.packages("rstan", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))
install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos"))
```

Part V

UNIT 5

4 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

```
install.packages("rstan", repos = c("https://mc-stan.org/r-packages/", getOption("repos")))
install.packages("cmdstanr", repos = c("https://mc-stan.org/r-packages/", getOption("repos"))
```

Part VI

UNIT 6

Stress, Diet, and Plasma Acids

```
library(cmdstanr)
```

This example introduces tracking of deterministic variables and shows how to recreate the BUGS step function in Stan.

It is adapted from Unit 6: stressacids.odc.

Associated lecture video: Unit 6 lesson 5

Problem Statement

In the study Interrelationships Between Stress, Dietary Intake, and Plasma Ascorbic Acid During Pregnancy conducted at the Virginia Polytechnic Institute and State University, the plasma ascorbic acid levels of pregnant women were compared for smokers versus non-smokers. Thirty-two women in the last three months of pregnancy, free of major health disorders, and ranging in age from 15 to 32 years were selected for the study. Prior to the collection of 20 ml of blood, the participants were told to avoid breakfast, forego their vitamin supplements, and avoid foods high in ascorbic acid content. From the blood samples, the plasma ascorbic acid values of each subject were determined in milligrams per 100 milliliters.

I start with the data pasted from `stressacids.odc`, then create one list for smokers and one for nonsmokers.

```
plasma <- c(0.97, 0.72, 1.00, 0.81, 0.62, 1.32, 1.24, 0.99, 0.90, 0.74,
          0.88, 0.94, 1.06, 0.86, 0.85, 0.58, 0.57, 0.64, 0.98, 1.09,
          0.92, 0.78, 1.24, 1.18, 0.48, 0.71, 0.98, 0.68, 1.18, 1.36,
          0.78, 1.64)

smo <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2)
```

```r
nonsmoker_index <- which(smo == 1)
plasma_smokers <- plasma[-nonsmoker_index]
plasma_nonsmokers <- plasma[nonsmoker_index]
```

BUGS `step` function

BUGS defines the step function like this:

$$step(e) = \begin{cases} 1, & e \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Stan follows what you would expect in a programming language. We implement this like:

```
e >= 0 ? 1 : 0;
```

How do I track non-random variables in Stan?

One nice thing about BUGS is you can easily track both deterministic and non-deterministic variables while sampling. For Stan, you add the variable to the `generated quantities` block.

```r
mod <- cmdstan_model(stress_diet_stan)
```

Stress, Diet, and Plasma Acids Model

```
data {
  int<lower=0> N_smoker;
  int<lower=0> N_nonsmoker;
  vector[N_smoker] plasma_smoker;
  vector[N_nonsmoker] plasma_nonsmoker;
}

parameters {
  real<lower=0> tau_nonsmoker;
  real mu_nonsmoker;
  real<lower=0> tau_smoker;
  real mu_smoker;
}

model {
```

```
  tau_nonsmoker ~ gamma(0.0001, 0.0001);
  tau_smoker ~ gamma(0.0001, 0.0001);
  mu_nonsmoker ~ normal(0, 100); // equivalent to BUGS tau = 0.0001
  mu_smoker ~ normal(0, 100);

  plasma_smoker ~ normal(mu_smoker, 1 / sqrt(tau_smoker));
  plasma_nonsmoker ~ normal(mu_nonsmoker, 1 / sqrt(tau_nonsmoker));
}

generated quantities {
  real testmu;
  real r;

  testmu = (mu_smoker >= mu_nonsmoker ? 1 : 0);
  r = tau_nonsmoker / tau_smoker;
}
```

Sampling

Let us prepare the data to be passed into sample

```
input_data <- list(N_smoker=length(plasma_smokers),
                   N_nonsmoker=length(nonsmoker_index),
                   plasma_smoker=plasma_smokers,
                   plasma_nonsmoker=plasma_nonsmokers
)
```

Now that we have the data to pass into our sampling, let's proceed.

```
fit <- mod$sample(
  data = input_data,
  seed = 123,
  chains = 4,
  parallel_chains = 4,
  refresh = 500,   # print update every 500 iterations
  iter_warmup = 1000,
  iter_sampling = 5000
)
```

```
fit$summary()
```

```
# A tibble: 7 x 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__         27.8   28.1   1.52   1.31   24.9   29.6    1.00    7997.    9743.
2 tau_nonsmok~ 22.6   21.9   6.68   6.53   12.9   34.5    1.00   17854.   12231.
3 mu_nonsmoker  0.912  0.912 0.0449 0.0436  0.839  0.986  1.00   17455.   12780.
4 tau_smoker    6.53   5.92  3.50   3.20    2.03  13.1    1.00   13257.   10335.
5 mu_smoker     0.977  0.977 0.162  0.145   0.721  1.24   1.00   13512.    9947.
6 testmu        0.667  1     0.471  0       0      1      1.00   16274.      NA
7 r             4.83   3.72  4.27   2.19    1.43  11.7    1.00   14304.   11261.
```

These results are very similar to PyMC results.