Particle Based Samplers for MCMC

Jon-Paul Cavallaro

Monday 17 February 2020

Contents

Chapter 1

First chapter title goes here

Contains implementations of particle based sampling methods for model parameter estimation. Primarily an implementation of the Particle Metropolis within Gibbs sampler outlined in the paper available at https://arxiv.org/abs/1806.10089, it also contains code for covariance estimation and time varying models.

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports, e.g., a math equation $a^2 + b^2 = c^2$.

The **bookdown** package can be installed from CRAN or Github:

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading `#`.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.name/tinytex/.

Chapter 2

Introduction

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2020) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

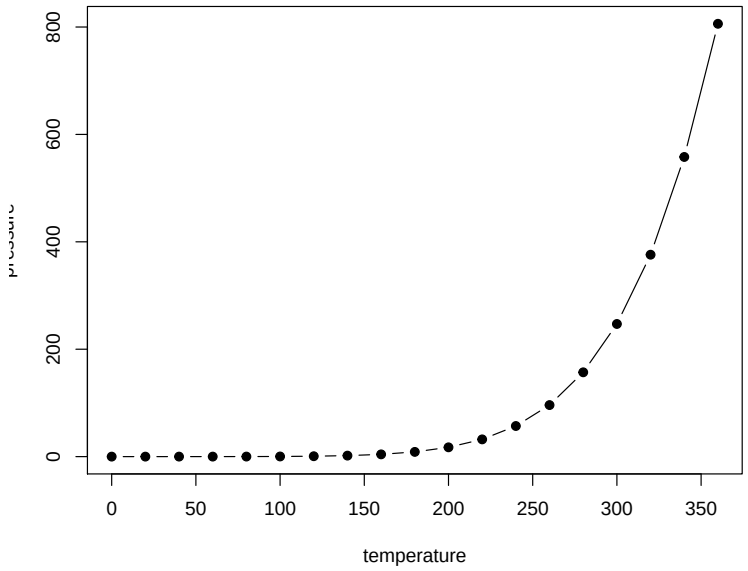


Figure 2.1: Here is a nice figure!

Chapter 3

Forstmann et al. Dataset

Here is an example dataset. We will use the dataset from Forstmann et al. (2008).

What packages required? Are they mentioned here? with r code? Version of R?

First we need to install the psamplers package

```
library(psamplers)
```

3.1 Description of Forstmann experiment

Forstmann et al looked at neural correlates of decision making under time pressure, with an aim to identify areas of the brain associated with speed-accuracy tradeoff. Imaging (fMRI) and behavioural data was collected; however, we'll look at the behavioural data from the decision-making task only. In terms of modelling the data, Forstmann expected to find differences in thresholds (direction?) for each of the three speed-emphasis conditions.

Table 3.1 shows the first ten trials from the Forstmann dataset (show to get an idea of the data structure?). Participants `(n = 19)` were asked to indicate whether a cloud of dots in a random-dot kinematogram (RDK) moved to the left or the right of the screen. The IV was a within-subject, speed-accuracy manipulation where, before each trial began, pariticipants were instructed to make their choice accurately `(condition = 1)`, with urgency`(condition = 3)`or were confronted with a neutral message `(condition = 2)`. Choice and response time data was collected. Choices were coded as correct `(correct = 2)` or incorrect `(correct = 1)` and response times `(rt)` were recorded in seconds. For more information about the design of the experiment please see the original paper.

Table 3.1: First 10 trials in Forstmann dataset. The 'forstmann' dataset is a data frame

subject	rt	correct	condition
1	0.4319	2	1
1	0.5015	2	3
1	0.3104	2	3
1	0.4809	2	1
1	0.3415	2	1
1	0.3465	2	2
1	0.3572	2	2
1	0.4042	2	2
1	0.3866	2	2
1	0.3683	2	1

3.2 Using the sampler

In this example, we will use the Linear Ballistic Accumulator (LBA) Brown and Heathcote (2008). The LBA model parameters are: `b` threshold parameter (the evidence required to make a response), `v` is drift rate or average speed of evidence accumulation, `A` is the model's start point and `t0` is non-decision time. Do we need to mention `sv` here?

First, we create a vector of parameter names for the model `pars`

```
pars <- c("b1", "b2", "b3", "A", "v1", "v2", "t0")
```

For this dataset, we use 3 threshold parameters (`b1`, `b2`, and `b3`) because we expect threshold to vary as a function of speed-emphasis instruction. We include two drift rate parameters: `v1` for the incorrect accumulator and `v2` for the correct accumulator, a start point parameter `A` and a non-decision time `t0` parameter. Note: It is important that the parameters you list in the `pars` vector match the names and number of parameters you include in your log likelihood function. We've made a decision to set the `sv` to 1 to satisfy the scaling properties of the model, as such we haven't included the `sv` parameter in the `pars` vector.

In the new estimations paper pg 6 has an equation with a vector of individual parameters/random effects - include here?

Next we create a `priors` object; a list that contains two components: - Gavin needs to check this priors object stuff * `theta_mu` a vector containing the prior for model parameter means * `theta_sig` the prior covariance matrix for model parameters.

```
priors <- list(theta_mu = rep(0, length(pars)),
  theta_sig = diag(rep(1, length(pars)))
)
```

The next step is to source your log likelihood function/script. This must be called before you create the sampler object in the next step.

```
source(file = "yourLogLikelihoodFile.R")
```

We need to supply LBA_loglike file

Next step is to initialise the `sampler` object.

```
sampler <- pmwgs(
  data = data0,
  pars = pars,
  prior = priors,
  ll_func = lba_loglike
)
```

The `pmwgs` function takes a set of arguments listed below and returns a list containing the required components for performing the particle metropolis within Gibbs steps.

- `data =` your data (a data frame containing a column for participants called 'subject')
- `pars =` the model parameters to be used (`pars` vector specified above)
- `prior =` the priors to be used (`priors` object)
- `ll_func =` name of log likelihood function you've sourced above.

You have the option to set model start points. We have specified sensible start points for the Forstmann dataset. If you chose not to specify start points, the sampler will randomly sample points from the prior distribution.

```
start_points <- list(
  mu = c(.2, .2, .2, .4, .3, 1.3, -2),
  sig2 = diag(rep(.01, length(pars)))
)
```

The `start_points` object contains two vectors: `mu` a vector of start points for the mu of each model parameter and a `sig2` vector containing the start points of the covariance matrix of covariance between model parameters.

Okay - now we are ready to run the sampler.

```
sampler <- init(sampler, theta_mu = start_points$mu,
                theta_sig = start_points$sig2)
```

Here we are using the `init` function to generate initial start points for the random effects and storing them in `sampler` object. First we pass the `sampler` object from above that includes our data, parameters, priors and log likelihood function. If we decided to specify our own start points (as above), we would include the `theta_mu` and `theta_sig` arguments.

Now we can run the sampler using the `run_stage` function. First we pass our `sampler` object that includes the parameters. The `stage =` argument specifies the sampling stage. `iter =` is the number of iterations of the sampling stage. `particles =` is the number of particles generated on each iteration. It is optional to include the `iter =` and `particles =` arguments. If these are not included, default values of 1000 are used. The number of iterations you choose for your burn in stage is similar to choices made when running deMCMC, however, this varies depending on the time the model takes to reach the 'real' posterior space.

First we run our burn in stage by setting `stage =` to `"burn"`

```
burned <- run_stage(sampler, stage = "burn", iter = 500, particles = 1000)
```

Here we have set iterations to be 500, which may take some time.

```
adapted <- run_stage(burned, stage = "adapt")
```

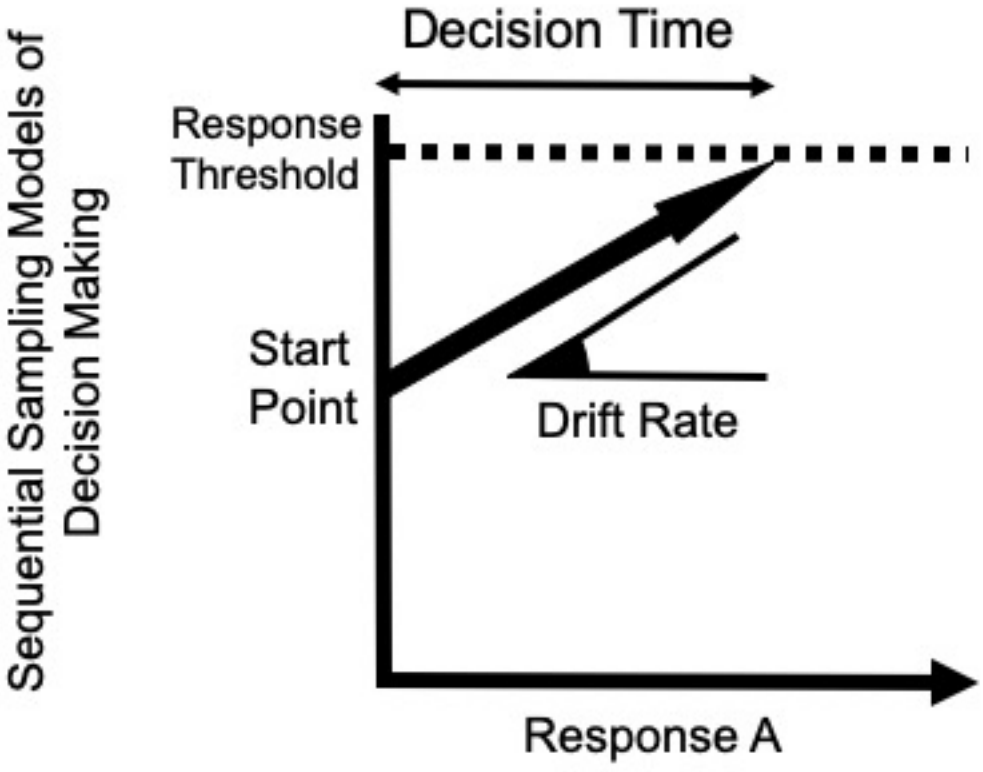
Now we run our adaptation stage by setting `stage = "adapt"` Because we have not specified number of iterations or particles, the sampler will use the default values of 1000 for each of these arguments. N.B. The sampler will quit adaptation stage after 20 unique values have been accepted for each subject. This means adaptation may not use all 1000 iterations.

```
sampled <- run_stage(adapted, stage = "sample", iter = 100, particles = 100)
```

At the start of the `sampled` stage, the sampler object will create a 'proposal' distribution for each subject's random effects using a conditional multi-variate normal. This proposal distribution is then used to efficiently generate new particles for each subject which means we can reduce the number of particles on each iteration whilst still achieving acceptance rates.

Chapter 4

The Linear Ballistic Aclimatiser



and caution. A basic framework for a simple two choice paradigm is shown. The two accumulators "race" by accumulating evidence for either decision before reaching the response threshold. The first accumulator to reach the threshold triggers the associated decision.

Chapter 5

Applications

Some *significant* applications are demonstrated in this chapter.

5.1 Example one

5.2 Example two

Chapter 6

Final Words

Nice book - dickheads.

Bibliography

Brown, S. D. and Heathcote, A. (2008). The simplest complete model of choice response time: Linear ballistic accumulation. *Cognitive psychology*, 57(3):153–178.

Forstmann, B. U., Dutilh, G., Brown, S., Neumann, J., Von Cramon, D. Y., Ridderinkhof, K. R., and Wagenmakers, E.-J. (2008). Striatum and pre-sma facilitate decision-making under time pressure. *Proceedings of the National Academy of Sciences*, 105(45):17538–17542.

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.17.1.