Doing Bayesian Data Analysis in Julia using Turing.jl

Kianté Fernandez

8/25/2022

Table of contents

What and why

Kruschke began his text with, "This book explains how to actually do Bayesian data analysis, by real people (like you), for realistic data (like yours)." In the same way, this project is designed to help those real people do Bayesian data analysis. My contribution is converting Kruschke's JAGS and Stan code for use in another probabilistic programming framework,`Turing.jl`, which makes it easier to fit Bayesian regression models in Julia (Ge, Xu, and Ghahramani (2018)) using a number of samplers. I also prefer plotting and data wrangling with the packages from `Plots.jl`(Bezanson et al. (2017)). So we'll be using those methods, too.

This ebook is not meant to stand alone. It's a supplement to the second edition of Kruschke (2015) Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Please give the source material some love.

Julia setup

To follow along with this guide, you'll need some software. Download and install Julia by following the instructions at https://julialang.org/downloads/. The Getting Started page has in depth instructions that can help.

Version 0.0.1.

I am just starting this project. I plan to have a complete draft including material from all the chapters in Kruschke's text by January 2023

1 What's in This Book (Read This First!)

1.1 Gimme feedback (be polite)

I am not a statistician and have no formal computer science background. I am in the process of learning the Julia programming language (part of the goal of this project!). I am currently a Ph.D. student in psychology. I have been mostly an R and MATLAB user and started learning Bayesian statistics around 2019. My code will likely be "bad" as I get the hang of things. I have much to learn from the Julia community and thus encourage folk to reach out with suggestions on how to improve my code. If you'd like to learn more about me, you can find my website at https://www.kiantefernandez.com/.

1.2 Thank you!

A. Solomon Kurz really inspired this project. He has published multiple accessible introductory content on applied Bayesian analysis, complementing many of the books that taught me Bayesian statistics. I benefitted greatly from his free content. Go find him at: https://solomonkurz.netlify.com.

2 Introduction: Credibility, Models, and Parameters

2.1 Bayesian inference is reallocation of credibility across possibilities

To make Figure 2.1 we need data. We will create some synthetic data and store it in using `DataFrames.jl`

```julia
using DataFrames

function expand_grid(; iters...)
    var_names = collect(keys(iters))
    var_itr = [1:length(x) for x in iters.data]
    var_ix = vcat([collect(x)' for x in Iterators.product(var_itr...)]...)
    out = DataFrame()
    for i = 1:length(var_names)
        out[:,var_names[i]] = collect(iters[i])[var_ix[:,i]]
    end
    return out
end

d = expand_grid(iteration=1:3, Possibilities=["a", "b","c", "d"], stage = ["a", "b"])

d2 =DataFrame(Credibility =[fill(.25,4); 0; fill(1/3,3); 0; fill(1/3,3);0;.5;0;0.5;0;.5;0;

sort!(d, [:iteration])
d.Credibility = d2.Credibility
```

```
Warning: use values(kwargs) and keys(kwargs) instead of kwargs.data and kwargs.itr
  caller = expand_grid(; iters::Base.Pairs{Symbol, AbstractVector, Tuple{Symbol, Symbol, Sy
@ Main ./In[2]:5
```

```
24-element Vector{Float64}:
 0.25
 0.25
 0.25
 0.25
 0.0
 0.3333333333333333
 0.3333333333333333
 0.3333333333333333
 0.0
 0.3333333333333333
 0.3333333333333333
 0.3333333333333333
 0.0
 0.5
 0.0
 0.5
 0.0
 0.5
 0.0
 0.5
 0.0
 0.0
 0.0
 1.0
```

Here we are defining a function that creates combinations and then I just bind that we the Credibility values we will use for plotting.

We can take a look at the top few rows of the data with the `first()` function.

```
first(d,5)
```

	iteration	Possibilities	stage	Credibility
	Int64	String	String	Float64
1	1	a	a	0.25
2	1	b	a	0.25
3	1	c	a	0.25
4	1	d	a	0.25
5	1	a	b	0.0

Now lets plot our version of Figure 2.1: