One Bayesian Workflow:

Below I outline a basic Bayesian workflow for someone using `Stan`. This is a short overview of a much deeper topic. See Bayesian statistics and modelling for a little more depth, including how to develop priors and your basic model formulation.

Note that you might need to start simple and build up to get all these steps to work, but this is my recommended approach to the basic workflow.

1. Check your code. Write down your model – I recommend as basic math, then write it in `Stan` and simulate one set of test data (often in `R`). I recommend you do this first because it will flesh out any issues in your simulated data (`R`) code, which you need right away (but you don't need the `Stan` code until step 3).
   (a) Write your `Stan` code.
   (b) In `R`, write simulated data where you **write out all your parameters**, write x and generate y from your model. So, if I am doing simple regression ($y = mx + b$, with error ϵ) then I would have to assign values to m, b and ϵ and I would generate a vector of x values, then I would simulate y from those values.
   (c) Run your `Stan` code on your simulated data. Check that your `Stan` code returns your model parameters, if not, check your code, set your error lower and/or sample size higher. Keep checking until your `Stan` output matches your parameters (note check your `Stan` code and your simulated data code) and you trust both.
2. Prior predictive checks: Check yo' priors.
   (a) Take your aforementioned `R` code and set up priors for each parameter (e.g., distributions for m, b, ϵ). **In contrast to above, where you just set each parameter to one value, here you you want to draw multiple values for each parameter and visually check the output.**
   (b) How do I check the output? Just like posterior predictive checks, this is up to you! At a minimum I recommend thinking of plots you will make with your model in the end (for publications) and plotting that given different prior values.
   (c) Your goal here is to check that your priors are reasonable, if they are not, adjust them.
   (d) Unlike in Step 1, you do **not** need to run `Stan`—you have your parameters so you can just simulate data from them, and then plot, examine etc.. No `Stan` at this step.
3. Now you can run your model on your real data!
   (a) Check the output, if you have divergent transitions, you need to re-parameterize your model (may tried a non-centered model or such).
   (b) I recommend ShinyStan here.
4. Posterior predictive checks: How good or bad does your model do compared to your data?

(a) Grab the parameter values from your fitted `Stan` model. In posterior predictive checks *these are the parameter values you use to simulate new data.*

(b) You can adapt your `R` code for simulating data above, but use your estimated parameters from your fitted model.

(c) What should I look at? Ah, just like in prior predictive checks, you need to decide. Classic things are to look at the mean of 100 or so simulations of new data you generate versus your **real** data. Also try the SD. Plot things! Look at the distribution. If you use the generated quantities block in `Stan` ShinyStan will automatically generate a few plots but think hard about more.

(d) When does this get hard? In hierarchical models (and other models with hyperparameters) as you have multiple levels you can generate—you can use *your estimated parameters from your fitted model at all levels or generate your lower-level parameters* (for example, you can generate species means using your species μ and σ). You have to think about what you want your model to predict.

(e) See how it is—do you see obvious problems caused by the distribution you selected or a grouping factor you're missing? If so, add it. And go back to Step 1.

(f) (No `Stan` at this step.)