Presenting Bayesian model output

Johannes Karreth

Applied Introduction to Bayesian Data Analysis

The purpose of this tutorial is to show you some options to work with and efficiently present output from Bayesian models in article manuscripts: regression tables, regression plots, marginal effects, predicted probabilities, and dot plots from factor models with uncertainty.

R code for each of the exercises is provided on `https://github.com/jkarreth/Bayes` and `http://www.jkarreth.net/bayes-cph.html`. Almost all examples in this tutorial also show up in my presentation slides. You can reproduce what I do in the slides by working through the R code associated with each slide set, posted on the workshop website. Don't hesitate to contact me if you have any questions or run into problems.

The tasks below assume you have access to data and models you can use for postestimation. If you don't have models ready, I provide examples in my slides that you can reproduce.

1 Working with JAGS output

For most of the exercises in this tutorial (and your own analyses), you will want to work with JAGS output directly. One way to get this output in a form that you can use for the applications below is to follow these steps:

1. Convert your JAGS/BUGS output into an MCMC list

   - If you work with JAGS from the command line, or with WinBUGS or OpenBUGS, read the output into R using the coda package:
     - Fit your model in WinBUGS/OpenBUGS/JAGS, and save the coda files in your working directory.
     - Name these files and use a common stem, for instance `mymodel_chain1.txt`, `mymodel_chain2.txt`, and `mymodel_index.txt`.
     - In R, load the coda package:

       ```
       > library(coda)
       ```

     - Now, read your chain and index files into R.
     - Set your working directory:

       ```
       > setwd("/Users/johanneskarreth/R/Bayes/myproject")
       ```

     - Read in your JAGS output. This requires that the chains and index files (see above) are in your working directory.

       ```
       > chain1 <- read.coda("mymodel_chain1.txt", "mymodel_index.txt")
       > chain2 <- read.coda("mymodel_chain2.txt", "mymodel_index.txt")
       > mymodel.mcmc <- as.mcmc.list(list(chain1, chain2))
       ```

   - If you work with R2jags/R2WinBUGS/R2OpenBUGS, you can convert your jags/bugs object `mymodel.fit` into an MCMC list using:
     - `> mymodel.mcmc <- as.mcmc(mymodel.fit)`

2. Convert the mcmc list into a matrix and then a data frame:

   - ```
     > mymodel.mat <- as.matrix(mymodel.mcmc)
     > mymodel.dat <- as.data.frame(mymodel.mat)
     ```
     ```

- In this matrix/dataframe, each row contains one iteration from your samples, and each column identifies one parameter (such as regression coefficients)

3. If necessary, use the `grep()` function to easily extract the columns you will need for post processing. For instance, if you monitored a variety of parameters, but need only regression coefficients, which you named `beta[1]`, `beta[2]`, etc., you can do the following:

```
> mymodel.betas <- mymodel.dat[, grep("beta[", colnames(mymodel.dat), fixed=T)]
```

2 Regression tables

Bayesian variants of the regression model don't differ much in their interpretation from frequentist models—aside from the interpretation of uncertainty. Hence, if you want to present a regression table in your paper, a "Bayesian" regression table will look very similar to the "frequentist" regression tables you are all familiar with. For Bayesian models, you want to display the standard deviation of the posterior estimates in lieu of coefficient standard errors (Table 1). Alternatively, you may choose to display the 95% credible interval besides (or below) the posterior mean (Table 2).

Table 1: Posterior estimates: Moral integration of American cities.

	Posterior mean	Posterior SD
Intercept	19.69	1.24
Diversity	-0.11	0.02
Mobility	-0.19	0.04
N	43	

Table 2: Posterior estimates: Moral integration of American cities.

	Posterior mean	95% CI
Intercept	19.69	[17.21; 22.14]
Diversity	-0.11	[-0.14; -0.07]
Mobility	-0.19	[-0.26; -0.12]
N	43	

Exercise 1. Estimate a quick Bayesian linear model. Present your model output in a regression table with either (a) standard deviations of or (b) 95% credible intervals of the posterior mean for each coefficient. When comparing different (nested) models, consider including a goodness-of-fit measure in your table.

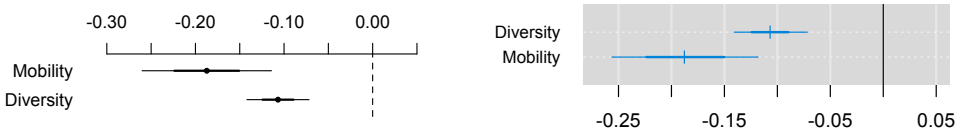
- If you use R2jags or R2WinBUGS, you can extract these quantities from the model or MCMC objects and use `xtable` to process them for LaTeX or HTML (for use in Word documents).
  - You can access the summary table from an R2jags or R2WinBUGS object (e.g., named `angell.fit`) and then process it with xtable:

```
> angell.fit$BUGSoutput$summary[, c(1, 2, 3, 7)]
>
> ##                mean         sd        2.5%        97.5%
> ## alpha     19.6053268 1.23238698  17.0721902  21.94372377
> ## beta1     -0.1066131 0.01738493  -0.1416944  -0.07276382
> ## beta2     -0.1842899 0.03744703  -0.2568409  -0.10719614
> ## deviance 192.6914225 2.95255745 188.9477266 200.11577668
>
> xtable(angell.fit$BUGSoutput$summary[, c(1, 2, 3, 7)])
```

  - The file `regression.table.R` on `https://github.com/jkarreth/Bayes` shows an additional convenient option for making tables.

3 Regression dot plots

Coefficient dot plots (or caterpillar plots) are a more intuitive way to present regression results. The reader can evaluate the size and significance of relationships more easily, and if the coefficients are on a comparable scale, dot plots make it more convenient to compare the size of relationships or effects. If you search the web for regression coefficient plots, you will find a variety of solutions. As



(a) Using the `coefplot` command

(b) Using the `caterplot` command

Figure 1: Regression coefficient plots.

canned functions, R offers the `coefplot` command in the arm package, the `caterplot` command in the mcmcplots package, and a few more. And, you can easily write script for your own dot plot that you can customize. One such example is at `https://github.com/jkarreth/JKcoefggplot`.

Exercise 2. Present your model output as a regression coefficient dot plot, including 95% credible intervals.

- Make sure that the output is stored in R as an MCMC object. Now, you can use the `caterplot` command in the package mcmcplots to produce a dot plot with your regression coefficients.
- If you would like to generate your own plot, create vectors for the mean posterior estimates, their standard deviation, and the coefficient/variable names you would like to use in your plot.
- Now, you can make a dot plot by hand, or by using the coefplot package.
- The file `regression.dotplot.R` on `https://github.com/jkarreth/Bayes` has step-by-step code for these two methods.
- You can also try and use the `JKcoefggplot` coefficient plot function on my Github repository for a more customizable plot.

4 Conditional effects: interactions

Conditional effects from interaction terms cannot be adequately represented in a standard regression table (Brambor, Clark, and Golder, 2006). In a frequentist regression model, you can calculate the conditional effect of $x1$ across the range of a moderator $x2$ by using the following equations for the effect and its variance:

$$\frac{\partial y}{\partial x1} = \widehat{\beta_1} + \widehat{\beta_3} \times x2 \tag{1}$$

$$\sigma^2 = \text{var}(\widehat{\beta_1}) + x2 \times \text{var}(\widehat{\beta_2}) + 2 \times x2 \times \text{cov}(\widehat{\beta_1}\widehat{\beta_3}) \tag{2}$$

In a Bayesian model, you can take advantage of the posterior variance to calculate the variance of the effect. The results should be virtually identical.

Exercise 3. Estimate a regression model using the example dataset `InteractionEx` from Dave Armstrong's DAMisc package, of the form:

$$y_i = \alpha + \beta_1 x1_i + \beta_2 x2_i + \beta_3 x3_i + \varepsilon_i \tag{3}$$

and present the effect of $x1$ across the range of $x2$.

- The file `interaction.instructions.R` on `https://github.com/jkarreth/Bayes` has code with solutions for a frequentist and a Bayesian interaction model.

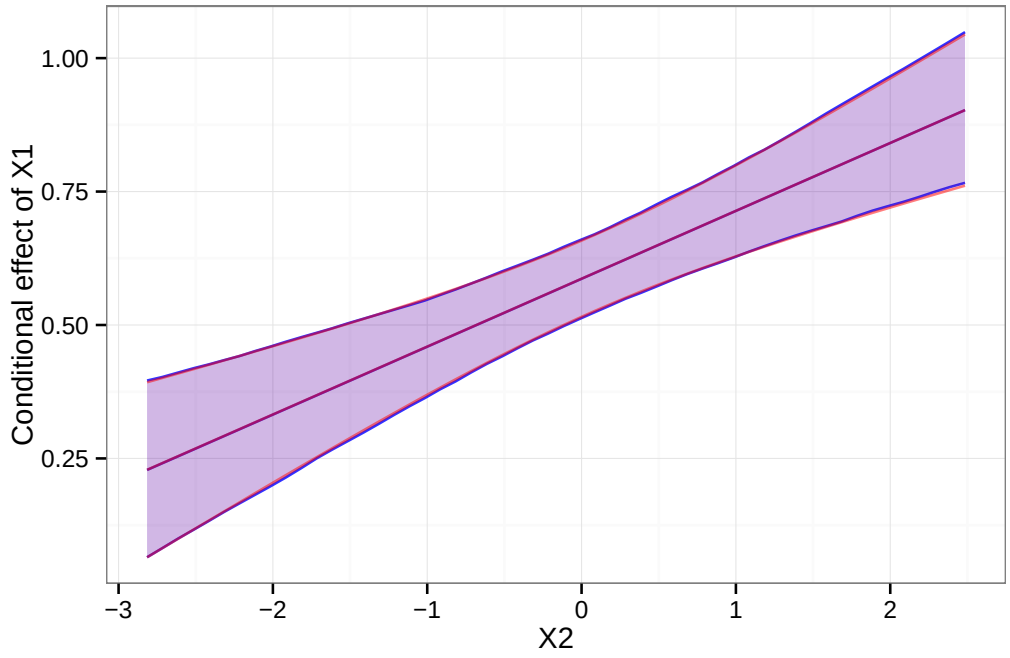


Figure 2: Conditional effects for x1 across the range of x2. Bayesian results: blue; frequentist results: red. The purple overlap indicates virtual identity of the two results.

5 Predicted probabilities

For nonlinear models, predicted probabilities are usually the most intuitive option to present interpretable results to readers. The most straightforward way to use them is to plot the predicted probability of the outcome across the simulated range of a predictor. To do this using the output of a Bayesian model for binary outcomes, follow the steps below.

Exercise 4. Estimate a Bayesian binary logit, ordered logit, or multinomial logit model and present predicted probabilities across the range of a substantively interesting covariate.

- Import the chains containing the coefficients from your BUGS/JAGS model, after monitoring the posterior coefficient distributions.
- Generate a vector with the simulated range of x_1 (here, age).
- Generate vectors set at desired values of the other covariates, with the same length as the simulated predictor from the previous step. Don't forget to generate a vector of 1s for the intercept.
- Make a dataset/dataframe with the simulated values.
- Multiply x_1 by the coefficients from your BUGS or JAGS output.
- Transform the linear predictions to probabilities.
- Generate mean predictions over the n (from BUGS/JAGS iterations) sampled values of the coefficients.
- Generate credible intervals, using the standard deviation (or percentiles) from your BUGS or JAGS output.
- Plot mean probability against the full (simulated) range of x_1.
- Add credible intervals.

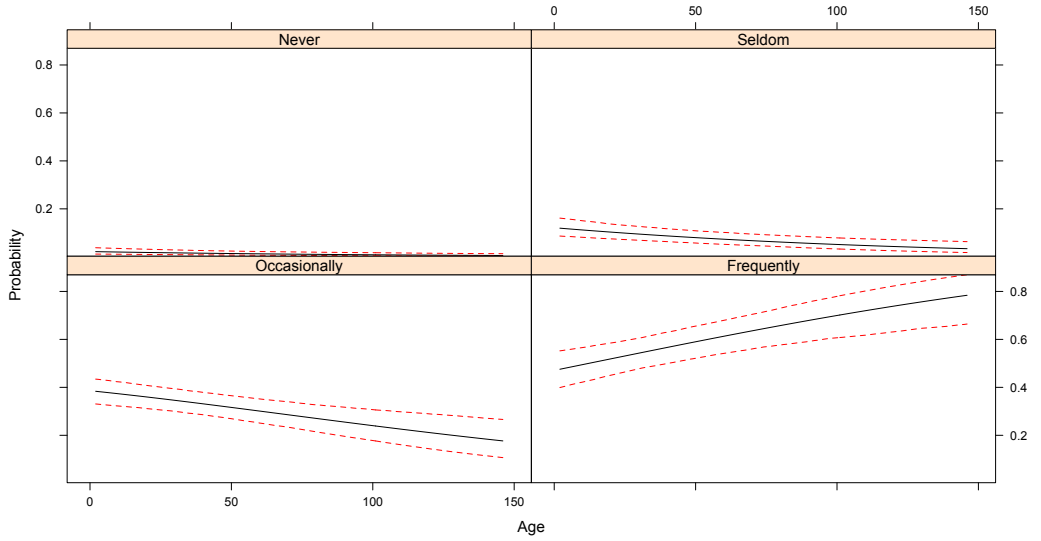


Figure 3: Example for plotting predicted probabilities for ordinal/categorical outcomes (here: on an ordinal scale from 1 to 4) across the (simulated) range of an independent variable (here: Age).

- The files `logit.pp.plot.instructions.R` and `ologit.pp.plot.instructions.R` on `https://github.com/jkarreth/Bayes` show how this can be done.

6 Proportional reduction of error

One measure of fit for models on categorical data (logit, ordered logit, multinomial logit) is to compare the predicted categories for each observation to the modal prediction of the model (i.e. a model that predicts for each observation the category matching the modal outcome in the observed data). When you fit a Bayesian model, you can obtain a posterior distribution for this measure of proportional reduction of error (PRE). On how to calculate the PRE, read (for instance) section 4 in Herron (1999). Comparing the distributions of the PREs of nested models than allows introducing (un)certainty to making statements on comparing model fits.

Exercise 5. Fit a binary, ordered, or multinomial logit model and assess model fit through the posterior distribution of the PRE. The file `ologit.pp.instructions.R` on `https://github.com/jkarreth/Bayes` shows how this can be done.

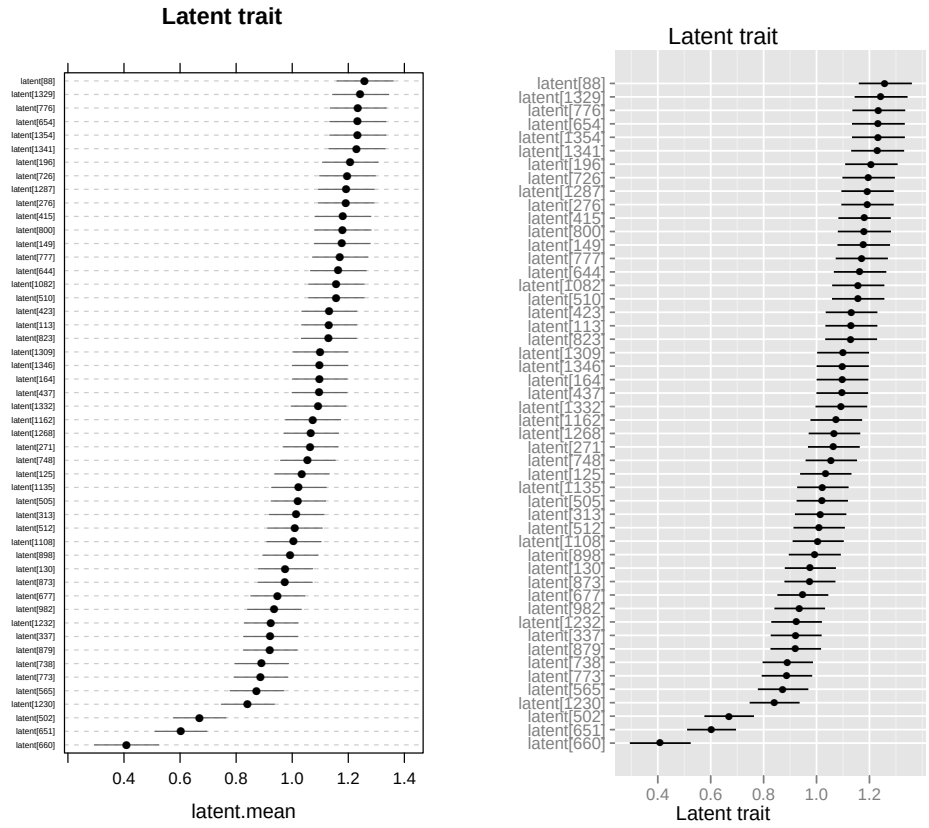
7 Factor dot plots

One key advantage of using Bayesian factor analysis is that you will obtain uncertainty estimates around the estimated factor scores. This is useful to ascertain differences between subjects' estimated factor scores: how meaningful are these differences really? For an illustration, see Treier and Jackman (2008).

Exercise 6. After estimating a Bayesian factor model, create a dot plot that graphically presents the uncertainty (via 95% credible intervals) around your estimated factor scores. The file `factor.dotplot.R` on `https://github.com/jkarreth/Bayes` shows how this can be done.

- First, extract the posterior distribution of your predicted factor scores from your BUGS/JAGS output.
- Identify the quantities of interest (mean, lower and upper bound of the credible interval) in this output.

- Arrange your output in a meaningful order (most likely, from low to high predicted factor scores; or by other qualities of interest, such as alphabetic observation IDs, if applicable).
- Produce a factor dot plot as in Figure 4.
- The file `factor.dotplot.R` on `https://github.com/jkarreth/Bayes` shows also shows how this can be done.



(a) Using the lattice package

(b) Using the ggplot2 package

Figure 4: Dot plots for factor scores.

Hint: The dot plot becomes cluttered as the number of observations increases, so be sure to inspect your data first. For an example with many observations, I used the following code to only plot 50 random draws from the 1,428 observations in the data for illustrative purposes:

```
> dat <- dat[sample(1:nrow(dat), 50, replace=FALSE),]
```

In your applied work, it will usually only make sense to produce dot plots for up to 100 observations, or more if split into several plots. If you have time-series or panel data, it might make sense to only present plots for one panel or time unit, or split the observations over several plots.

References

Brambor, Thomas, William R. Clark, and Matt Golder. 2006. "Understanding Interaction Models: Improving Empirical Analyses." *Political Analysis* 14 (1): 63–82.

Herron, Michael C. 1999. "Postestimation Uncertainty in Limited Dependent Variable Models." *Political Analysis* 8 (1): 83–98.

Treier, Shawn, and Simon Jackman. 2008. "Democracy as a Latent Variable." *American Journal of Political Science* 52 (1): 201–217.