FILES

The following exercises are intended to familiarize you with fitting simple Bayesian models using MCMC. This document accompanies the following scripts (R language):

* **bayes\_shortcourse\_simdata.R:** used by all other programs to simulate a simple dataset
* **bayes\_shortcourse\_ex1\_indpriors.R:** Bayesian linear regression with independent coefficient priors
* **bayes\_shortcourse\_ex2\_hierarchical.R:** Bayesian linear regression with hierarchical “shared” coefficient priors
* **bayes\_shortcourse\_ex3\_mixture.R:** Bayesian linear regression with coefficient mixture priors
* **bayes\_shortcourse\_ex4\_bma\_gcomp.R:** Bayesian linear regression and g-computation with Bayesian model averaging/selection (BMA) priors

R programming note: MCMC and the data simulation steps both use random number seeds. For maximum comparability with your peers and the key, please run the “set.seed(…)” lines of code (2 per program) at the same time as you generate the data or fit the models via MCMC. To check whether you are doing this reproducibly, run the same model twice in a row to verify that you get the same answer each time – if not, then you are not setting the seed each time.

EXERCISES

1. Download these files from github as a zip file via [direct link](https://github.com/alexpkeil1/BayesBootcamp/archive/refs/heads/main.zip) or via the instructions here (no github account needed): <https://github.com/alexpkeil1/BayesBootcamp#readme>, or “clone” the repository if you are already familiar with github.
   1. Unzip the file in your Downloads folder (not required, but easier)
   2. Change the name of the unzipped folder to BayesBootcamp (not required, but easier)
2. Independent priors (bayes\_shortcourse\_ex1\_indpriors.R)
   1. *Adapt pathnames:* If needed, adapt the pathnames on lines 15 (Windows) or 18 (MacOS/Linux) of **bayes\_shortcourse\_ex1\_indpriors** to point to the directory “BayesBootcamp.”
   2. *Simulate data:* Run lines 20 and 21 of the program, which will call in the program “bayes\_shortcourse\_simdata.R.” Ensure that this has worked by running lines 24 and 25. Use “head(data1)” in R to look at the top 6 rows of the resulting simulated data.
   3. *Check Bayes model code:* Examine the JAGS code that is contained in the R function labeled “b.ind” (line 37).
      1. What is the name of the exposure that corresponds to the parameter “b1” in the model? **\_V1\_\_**
      2. What are the mean \_\_**0.0**\_\_ and variance \_\_**100**\_\_ of the *prior* normal distribution for the “b1” parameter. (Keep in mind that the JAGS function “dnorm” is parameterized as “dnorm(mean, 1/variance).”
   4. *Check MCMC specification:* using the call to the “jags” function (called directly from the package with “R2jags::jags”), what are:
      1. The number of independent Markov Chains? C=\_\_**3**\_\_
      2. The number of total iterations/simulated samples per chain? I=\_\_**5000**\_\_\_
      3. How many of these iterations are removed before making inference (i.e. what is the “burn in”?) B=\_\_**2500**\_\_
      4. How many total iterations/simulated samples will you have available for inference? C\*(I-B)=\_\_**7500**\_\_
   5. *Run the model:* Run the code from lines 67-69 (this will run the MCMC sampler for all of the iterations specified in the previous step).
   6. *Check the posterior means/standard deviations:*
      1. What is the posterior mean of the “b1” parameter (mu.vect)? \_**0.471**\_\_\_
      2. Calculate normality based 95% Credible intervals by mu.vect +/- 1.96\*sd.vect. What are these?\_\_\_**0.052\_\_, \_\_0.890**\_\_\_
      3. Do the normality based 95% Credible intervals agree with the percentile based Credible intervals (within, say 1 significant digit)? \_\_**yes**\_\_
      4. Do the R-hat values suggest that convergence has been met? \_\_**yes**\_\_
      5. Run the following code to fit a standard linear regression model to the data:

lm(y~V1+V2+V3+V4+V5, data = data1)

The output is from a standard linear regression model fit via least squares. Does the coefficient estimate for V1 agree with the “b1” posterior mean from the Bayesian model (to 2 decimal places)? \_\_ **0.47 yes**\_\_

* + 1. Run the following code to get the posterior median of “b1”:

median(as.matrix(m1.output[,'b1']))

Is this similar to the posterior mean? **\_\_ 0.468 yes**\_\_

* 1. *Graphical diagnostics*
     1. Examine the autocorrelation plot (line 79) across one of the MCMC chains. Autocorrelation describes how correlated each sequential sample is with the last sample, and low autocorrelation in an otherwise good MCMC chain means that you have an “efficient” sampler and do not need too many samples to make valid inference. Do you see any patterns in the autocorrelation plot? \_\_**no – autocor drops quickly, looks good**\_\_\_
     2. Examine the kernel posterior density of the “b1” parameter (line 80 – the “densplot” function). Does the kernel posterior density suggest any problems with this parameter (e.g. multiple modes) or reasons why the posterior mean may not be a good representative value of the whole posterior (e.g. high skew)? \_\_**no – looks good and symmetric, close to normal**\_\_
     3. Examine the trace plot of “b1” for all three chains (line 81). A good trace plot is said to look like a fuzzy caterpillar with no discernable trends (e.g. it is not trending upward or downward). Does this look like a good trace plot? \_\_**yes – looks like “noise” and all chains are similar**\_\_
     4. Try re-fitting the model and set burn-in to 0. Now examine the trace plot again. Quick convergence can be observed on this plot by seeing how long it takes for the “fuzzy caterpillar” to move from it’s initial values at iteration 1 to moving randomly around the posterior mean. Does it seem like there is quick convergence in this model? \_\_**yes – moves very quickly to the center of the caterpillar (so we could probably reduce burnin)**\_\_\_

1. Hierarchical prior (bayes\_shortcourse\_ex2\_hierarchical.R)
   1. Be sure this file will run (check pathnames, etc.)
   2. *Check Bayes model code:* Examine the JAGS code that is contained in the R function labeled “b.shared” (line 37).
      1. Parameters “b1” – “b5” share a common prior mean, where the prior mean parameter is called “beta.m.” This can be expressed hierarchically via:

|  |
| --- |
|  |
|  |
|  |

* + 1. The parameter is the prior standard deviation for coefficients, and the posterior of can be interpreted as an estimate of heterogeneity of the model coefficients (i.e. if we expect that all of the exposures have similar effect sizes, then we would expect to be small). What is our prior distribution for \_\_**uniform(min=0, max=100)**\_\_\_\_\_ (hint: look up what the “dunif” function is in JAGS). Would you characterize this as a “strong” or “vague” prior? \_**vague**\_\_\_
  1. *Run the model*
  2. *Check the posterior means/standard deviations*
     1. Does R-hat look OK? \_yes\_\_
     2. Compare the “beta.m” posterior mean with the arithmetic mean of the “true” model coefficients used to create the data (i.e. beta1-beta5 in the call to the “bayes.sim” function). What do you notice? \_\_**beta.m = .246**, **mean true coefficient = 0.3\*4/5 = 0.24: these are close**\_\_\_
     3. Again run the following code to fit a standard linear regression model to the data:

lm(y~V1+V2+V3+V4+V5, data = data1)

The output is from a standard linear regression model fit via least squares. Does the coefficient estimate for V1 agree with the “b1” posterior mean from the Bayesian model (to 2 decimal places)? \_ **.384 (bayes) vs. 0.47407 (ols) – don’t quite agree**!\_\_\_

* + 1. You can get the posterior median of via median(as.matrix(m2.output[,'sigma.beta']))

How does this compare to the posterior mean? \_ **0.232 (median) vs. 0.290 (mean) – median not equal to mean, which suggests this is a skewed parameter**\_\_\_ (you can look at a posterior kernel density plot to confirm what this means).

1. Mixture prior (bayes\_shortcourse\_ex3\_mixture.R)
   1. Be sure this file will run (check pathnames, etc.)
   2. *Check Bayes model code:* Examine the JAGS code that is contained in the R function labeled “b.mix” (line 36). The comments in the code are helpful here, and you may wish to consult <https://pubmed.ncbi.nlm.nih.gov/17272963/> to understand this model better).
   3. *Run the model*
   4. *Check the posterior means/standard deviations*
      1. What do you notice about the “b[5]” parameter relative to the corresponding parameter in the prior two models? **\_\_ estimate is very close to null 0.000 and very precise (posterior sd = 0.002), other models were further from null and less precise**\_\_\_ (hint: you can quickly compare using print(bayes.m1); print(bayes.m2); print(bayes.m3))
      2. What do you notice about the variation of the posterior means of “b[1]”-“b[4]” relative to the other two models (i.e. are these closer to each other than in previous models?) \_\_**these estimates are all very close to 0.3 (within 0.005), whereas other models had more spread**\_ \_\_
      3. Bonus: Try changing the data generating step so that beta1-beta5 all have unique values (say 0.8, 0.6, 0.4, 0.2, 0.0). What happens? \_\_posterior means: .6,.6,.6,.2, 0 – **this forces all the estimated coefficients to be very similar even if the true coefficients are not. The DIC goes up, suggesting poor fit, trace plot for b[1] also looks wacky (more burn-in likely needed)**\_\_
   5. *Graphical diagnostics*
      1. Examine a trace plot for ‘b[1]’. What do you notice relative to other models? \_ **it’s very spiky – this isn’t necessarily bad but is expected with this kind of model where the distribution changes across iterations** \_\_\_
2. Bayesian model averaging + g-computation (bayes\_shortcourse\_ex4\_bma\_gcomp.R)
   1. Be sure this file will run (check pathnames, etc.)
   2. *Check Bayes model code:* Examine the JAGS code that is contained in the R function labeled “b.bma” (line 49). The comments in the code are helpful here, and you may wish to consult <https://pubmed.ncbi.nlm.nih.gov/33751055/> to understand this model better).
   3. *Run the model*
   4. *Check the posterior means/standard deviations*
      1. Is R-hat acceptable for the beta parameters? If you see any R-hat values for these parameters > 1.005, then re-run with 10,000 iterations per chain and re-check.
      2. The “posterior inclusion probability” (PIP) parameters characterize the posterior probability that an exposure has a non-null coefficient in the model. “pip[5]” corresponds to exposure V5. Does “pip[5]” agree with what you have learned from other models? \_\_**yes – a near-zero probability of inclusion in the model is closely equivalent to having a model coefficient equal to 0.0**\_\_
      3. Compare “b[5]” mu.vect and sd.vect with the “independent” prior model (model 1). Does the BMA model describe **MORE** or LESS (choose one) certainty than model 1 than that the true coefficient for V1 is null? \_\_\_ **mean, sd = -0.001, 0.007 in BMA, vs -0.017, 0.021 in independent priors (more precision and estimate is closer to 0 in BMA**)\_\_\_\_
      4. The “meandiff” parameter is a result of (a simple form) of Bayesian g-computation. This parameter is interpreted as the expected change in the outcome if we could change every units’ exposures (V1-V5) to the observed medians (relative to no change). This is an example of a joint effect of an exposure mixture. Based on this parameter, would you expect such an intervention to **INCREASE** or DECREASE (choose one) the outcome? \_\_ **the code (meandiff <- mean(muint) - mean(mu)) implies that positive values of this parameter indicate the outcome is higher under the intervention, so the estimate of .250 indicates that the intervention would increase outcome.** \_\_
   5. *Graphical diagnostics*
      1. Examine a trace plot for ‘b[1]’. What do you notice relative to other models? \_\_**it’s multi-modal with a mode near zero (similar to mixture model, but more density associated with b[1]=0 in the BMA model)\_\_**