General strategy for implementing the hierarchical Bayesian linear (BHL) model

The zip file (Code C1) contains the following files:

- ModelScript_HBL_SI (.odc or .txt): An OpenBUGS model script file that points OpenBUGS to the correct model file, loads the correct data, loads initial values for each MCMC chain, compiles the model, sets nodes (parameters) to monitor, and produces convergence plots and posterior statistics.
- ModelCode_HBL_SI (.odc or .txt): The actual OpenBUGS code that specifies the statistical and process models for the HBL model, with annotation.
- Datasets required by the HBL model: Dataset1_SI.txt, Dataset2a_SI.txt, and Dataset2b_SI.txt
- Starting values for three MCMC chains: Initials_a_S1.txt, Initials_b_S1.txt, Initials_c_S1.txt

Running the example code and data provided:

- One can simply evaluate the script files provided in order to run the models presented in the manuscript, using the same data used in the manuscript.

Strategies for implementing your own models:

- Make appropriate modifications to the code to reflect the experimental design.
- Try to follow similar formatting of data. Note that all categorical variables (e.g., treatment levels, dates, etc.) are coded as integer values, beginning at 1, which often serve as "indexing" variables in the model code.
- The trickiest part of implementing the models may be coming up with reasonable initial or starting values. We summarize our approach to generating the initial values used in our HBL model (see below).

Strategies for generating initials values for each chain.

- In our HBL model, the following parameters (nodes) require initial values (below), and we describe approaches to generating initial values for each. Please see one of the initials files (e.g., Initials_a_S1.txt) for proper formatting (list syntax as used by R) for scalars, vectors, and arrays (structures). Note that we just need to get rough, ball-park estimates of the potential range of values for each node, we do not have to conduct formal, rigorous statistical analyses to generate these initials.
    - Ca.ppm
    - CorrCO2.rep
    - Slope
    - const
    - eps.C
    - mu.Ca.ppm
    - mu.slope
    - tau.CO2
    - tau.Ca
    - tau.eps
    - tau.S

- **Ca.ppm**: Ca.ppm is the trt x veg x date initial [CO2]. One should have a fairly good guess of the initial (background) [CO2]. For example, for the ambient plots, we could randomly generate values for the associated Ca.ppm from, for example, Normal(370, 5), where 5 is the standard deviation (there's nothing special about choosing 5). For elevated plots, we could randomly generate values from, for example, Normal(600, 5). If more precise initials are required (e.g., do to simulation crashing during beginning MCMC iterations, or poor and/or slow convergence), we could use the time zero (t = 0 sec) data of chamber [CO2] and compute sample means for each trt x veg x date combination. Since each MCMC chain should be initialized with different values, we could then add a small amount of random error to these sample means, and provide these values as initials. Also, if "good" initials are provided for mu.Ca.ppm and tau.Ca (parent nodes of Ca.ppm), then you may not need to provide initials for Ca.ppm (have OpenBUGS or JAGS generate initials).
- **CorrCO2.rep**: These are the replicated data, and thus the size of this vector or array matches that of the observed data. One could either (1) not provide initials for this node, and let OpenBUGS or JAGS generate them, or (2) simply add a small amount of random noise to the corresponding observed chamber [CO2] values and use these values with error as initials for CorrCO2.rep.
- **Slope**: This is the slope of the [CO2] (ppm) versus time (sec) line, for each chamber session group. One can simply conduct an OLS regression of observed [CO2] versus time (sec) for each chamber session, get the slope estimate, add a small amount of error (e.g., using the std error of the slope estimate provided), and use these values as initials. Again, each chain should have a unique set of initial values for Slope. For "bad" chamber sessions that gave unrealistic slope estimates, one could replace these with the average (or ballpark average) of the slopes for the same treatment group, plus some random error.
- **const**: This is the redundant / latent parameter introduced by the parameter expansion trick. Randomly pick values between 0.9 and 1.1 for each chain (const is a vector of length 2, one for each CO2 treatment level).
- **eps.C**: This is the other redundant / latent parameter introduced by the parameter expansion trick, with the same dimensions as Ca0.ppm (here, # chamber sessions or groups). This is essentially an error term that describes the deviation of each group-level initial [CO2] value from the expected treatment x vegetation type x date initial [CO2] value. One could obtain initial estimates of eps.C by subtracting the observed time zero [CO2] value for each chamber from the expected treatment level [CO2] value (e.g., 370 vs 600 pm for ambient and elevated, respectively). Or, one could actually compute the average observed initial [CO2] for each trt x veg x date combination, and estimate the group-level deviation as group-level observed initial [CO2] minus the corresponding trt x veg x date sample mean. Add reasonable (but not too large) amount of random noise to each value to obtain unique initial values for each MCMC chain.
- **mu.Ca.ppm**: This is the overall, treatment-level initial [CO2], which varies by CO2 treatment level (ambient and elevate, so vector of length 2). Provide reasonable values based on knowledge of what these background [CO2] values should be; add some noise (within 20%) to provide unique initials for each chain.
- **mu.slope**: This is the treatment x veg x date-level slope of [CO2] (ppm) versus time (sec). One can get decent estimates of this by conducting an OLS linear regression of observed [CO2] versus time (sec) for data that are pooled by treatment x veg x date categories. Add some random noise (e.g., within about 10-20%) to get unique initials for each chain.
- **tau.CO2**: This is the precision (1/variance) associated with the residual observation error, which varies by global change treatment level. A decent estimate can be obtained by computing the sample variance for the initial (time zero) [CO2] values grouped by treatment level. Invert this to

get the precision, and add some noise (within 1 order of magnitude, favoring smaller values) to produce initials for each chain.

- **tau.Ca**: This is the precision (1/variance) associated with variance in the initial (background) [CO2] among dates and veg types within each CO2 treatment level; tau.Ca varies by CO2 treatment level (vector of two values). Follow the procedure above for tau.CO2, but only group data by elevated vs ambient CO2 levels to compute sample variance, convert to precision, and add some noise.
- **tau.eps**: This is the precision (1/variance) associated with the latent error term (eps.C) that was introduced by the parameter expansion. When providing initial values for const around 1, this precision is essentially equivalent to the among group (session) variance in the initial [CO2], which varies by CO2 treatment level (vector of two values). Use the same procedure to generating initials for tau.eps as done for tau.Ca.
- **tau.S**: This is the precision (1/variance) that describes the variation in the slope parameters among chamber sessions (groups), which varies by treatment level. Decent estimates can be obtained by computing the variance among the slope estimates obtained from the OLS regression conducted to get starting values for Slope (above). Group the Slope point estimates by treatment level, compute the sample variance of these, invert to get precision, and add some noise to create initial values.