1 Things to discuss

1.1 Jin's code

- https://github.com/ying531/MCMC-SymReg

1.2 The hacky implementation of LinearCoefs means that the first k samples are not proper

1.3 Error with how I was tracking the variances

- Each sample has one residual variance
- But each tree has their own linear coefficients variances
- For now hacky fix

basicstyle=code-fg,stringstyle=code-string,commentstyle=code-comment,keywordstyle=code-keyword,upquote=true,bac bg,frame=single,framesep=5pt,rulecolor=code-comment,framerule=1pt,language=julia,label= ,caption= ,captionpos=b,numbers=none i > k ? previous$_i$ $= i - k : previous_i = iold_a = chain.samples[previous_i].[:$ $_a]old_b = chain.samples[previous_i].[:{}_b]$

1.4 Acceptance ratio calculation

1.4.1 Error with transition probabilities

- I had MH reversed in my head
- Numerator transition proposal -> old
- Denominator transition old -> proposal

1.4.2 Standard MH

- On Jin's paper they don't use the probability of the LinearCoef
  - I understand not using the probability of jumping between coefficients
    - Because it is symmetric (or in this case just from the prior)
- But why not include the probability of each coefficient?

f(Θ | S)

- They don't use either the probability of the variances (residuals + LinearCoef)
  - I had already included it in the ratio calculation without RJMCMC because I hadn't noticed they omit it

p(Σ)

- Maybe I'm just misunderstanding the equations

1.4.3 With RJMCM

- I was confused with what operations were set notation or vector notation
- The papers/tutorials you sent me have been the best source
- I thought that some notation in Jin's paper were typos, but I think it makes sense now
  - Except 1 on shrinkage
- I need to refactor parts of the code to accommodate the procedure
- … And I need a bit of scaffolding with the Jacobian D:
  - O Jin's code they use $2^{\text{no. linear operators}}$
    - I think

1. Expansion
   (a) Θ is the set of old parameters
   (b) Θ^* is the set of new parameters
   (c) We sample variances
   (d) $\dim(u_\Theta) = \dim(\Theta)$
   (e) $\dim(u_n) = \dim(\Theta^*) - \dim(\Theta)$
   (f) We sample u_Θ and u_n from the prior $p(\Theta)$
   (g) $\Theta^* = (\text{mean}(\Theta, u_\Theta), u_n)$ (set addition)
   (h) $h(U) = \prod^{\forall(u_\Theta \cup u_n)} p(u)$
   (i) We keep $U^* = \frac{\Theta - u_\Theta}{2}$
      - For the jacobian ????

(a) Example

$$(\theta_1,\ \theta_2) \rightarrow (\theta_1^* \dots \theta_4^*) = j(\Theta, u_\Theta, u_n) = (\theta_1 + u_{\Theta 1},\ \theta_2 + u_{\Theta 2},\ u_{n1},\ u_{n2})$$

$$J = \left|\frac{\partial \Theta^*}{\partial(\Theta, U)}\right| = \left|\frac{\partial j(\Theta, U)}{\partial(\Theta, U)}\right| =$$

$$\begin{vmatrix} \frac{\partial \theta_1 + u_{\Theta 1}}{\partial \theta_1} & \frac{\partial \theta_1 + u_{\Theta 1}}{\partial \theta_2} & \frac{\partial \theta_1 + u_{\Theta 1}}{\partial X} & \frac{\partial \theta_1 + u_{\Theta 1}}{\partial X} \\ \frac{\partial \theta_2 + u_{\Theta 2}}{\partial \theta_1} & \frac{\partial \theta_2 + u_{\Theta 2}}{\partial \theta_2} & \frac{\partial \theta_2 + u_{\Theta 2}}{\partial X} & \frac{\partial \theta_2 + u_{\Theta 2}}{\partial X} \\ \frac{\partial u_{n1}}{\partial \theta_1} & \frac{\partial u_{n1}}{\partial \theta_2} & \frac{\partial u_{n1}}{\partial X} & \frac{\partial u_{n1}}{\partial X} \\ \frac{\partial u_{n2}}{\partial \theta_1} & \frac{\partial u_{n2}}{\partial \theta_2} & \frac{\partial u_{n2}}{\partial X} & \frac{\partial u_{n2}}{\partial X} \end{vmatrix}$$

- What are X?
  - Could be u_Θ
  - or u_n
  - or this doesn't make sense?

2. Shrinkage

(a) Θ is the set of old parameters

(b) We sample variances

(c) Θ_0 are the parameters we keep

(d) Θ_d are the parameters we discard

(e) We sample U of size $\dim(\Theta_0)$ from $N(0, \sigma^2_{a|b})$

(f) $\Theta^* = \Theta_0 + U$ (vector addition)

- $\theta_i^* = \theta_i + U_i$

(g) $h(U) = \prod^{\forall U} p(U_i)$

(h) We keep $U^* = (\Theta - U, \Theta_0)$ (set addition)

- For the jacobian ????

(a) Example

$$(\theta_1 \dots \theta_4) \rightarrow (\theta_1^*,\ \theta_2^*) = j(\Theta, U) = (\theta_1 + U_1,\ \theta_2 + U_2)$$

$$J = \left|\frac{\partial \Theta^*}{\partial(\Theta, U)}\right| = \left|\frac{\partial j(\Theta, U)}{\partial(\Theta, U)}\right| =$$

$$\begin{vmatrix} \frac{\partial \theta_1 + U_1}{\partial \theta_1} & \frac{\partial \theta_1 + U_1}{\partial \theta_2} & \frac{\partial \theta_1 + U_1}{\partial \theta_3} & \frac{\partial \theta_1 + U_1}{\partial \theta_4} \\ \frac{\partial \theta_2 + U_2}{\partial \theta_1} & \frac{\partial \theta_2 + U_2}{\partial \theta_2} & \frac{\partial \theta_2 + U_2}{\partial \theta_3} & \frac{\partial \theta_2 + U_2}{\partial \theta_4} \\ \dots & & & \\ \dots & & & \end{vmatrix}$$

- What are the other two rows?
  - The Us sampled?
    - Shouldn't them be on the denominators as well?
  - Or what happens with the U^*s?
  - Or this doesn't make sense?

$$
\begin{vmatrix}
\frac{\partial\theta_1+U_1}{\partial\theta_1} & \frac{\partial\theta_1+U_1}{\partial\theta_2} & \frac{\partial\theta_1+U_1}{\partial\theta_3} & \frac{\partial\theta_1+U_1}{\partial\theta_4} \\
\frac{\partial\theta_2+U_2}{\partial\theta_1} & \frac{\partial\theta_2+U_2}{\partial\theta_2} & \frac{\partial\theta_2+U_2}{\partial\theta_3} & \frac{\partial\theta_2+U_2}{\partial\theta_4} \\
\frac{\partial U_1}{\partial\theta_1} & \frac{\partial U_1}{\partial\theta_2} & \frac{\partial U_1}{\partial\theta_3} & \frac{\partial U_1}{\partial\theta_4} \\
\frac{\partial U_2}{\partial\theta_1} & \frac{\partial U_2}{\partial\theta_2} & \frac{\partial U_2}{\partial\theta_3} & \frac{\partial U_2}{\partial\theta_4}
\end{vmatrix}
$$

1.5 Type stability

basicstyle=code-fg,stringstyle=code-string,commentstyle=code-comment,keywordstyle=code-keyword,upquote=true,bac bg,frame=single,framesep=5pt,rulecolor=code-comment,framerule=1pt,language=julia,label= ,caption= ,captionpos=b,numbers=none @code$_{w}arntypeBayesianSR.step(chain, 1, 1, verbose = false) => Body :: UnionFloat64, BayesianSR.Sample$

- Why??
- If I force step(...)::Sample, the warning disappears (as expected)
- No performance differences
- FIXED: NaN return when optim ! failed.
- I believe all the program is type stable now.

1.6 ExprBugs

- There weren't any bugs :D
  - Couldn't reproduce them
  - I misunderstood how mutation, scope, and assigning(=) worked before when I thought there were bugs.
- (apart from the insert! one)
  - https://github.com/sisl/ExprRules.jl/pull/31

2 Notes