BAND stats

Matthew Plumlee (on behalf of BAND)

March, 16, 2021

Typical Bayesian calibration

Consider the classical model

$$y_i = f(x_i, \theta) + \varepsilon_i \text{ for } i = 1, \ldots, n$$

where x_i is seen but θ is not.

A vectorized version has

$$y = f(\theta) + \varepsilon, \text{where } \varepsilon \sim \text{MVN}(0, W).$$

Our posterior here has the form

$$p(\theta|y) \propto |W|^{-1/2} \exp\left(-\frac{1}{2}(y - f(\theta))^{\mathsf{T}} W^{-1}(y - f(\theta))\right) p(\theta).$$

W can include discrepancy, but be careful!

Tuo, R. and Wu, C.J., 2015. Efficient calibration for imperfect computer models. Annals of Statistics, 43(6), pp.2331-2352.

Plumlee, M., 2017. Bayesian calibration of inexact computer models. Journal of the American Statistical Association, 112(519), pp.1274-1285

Bayesian calibration with an emulator

Say that $f(\theta)$ is unknown, but has a prior distribution of a normal distribution with mean $\mu(\theta)$ and covariance $\Sigma(\theta)$.

An **emulator** is the predictive mean $\mu(\cdot)$ and covariance $\Sigma(\cdot)$ based on data from a computer experiment.

Our revised posterior is then

$$p(\theta|y) \propto |W + \Sigma(\theta)|^{-1/2} \exp\left(-\frac{1}{2}(y - \mu(\theta))^{\mathsf{T}}(W + \Sigma(\theta))^{-1}(y - \mu(\theta))\right) p(\theta).$$

$\Sigma(\theta)$ should include cross-correlation surrogation errors between xs

Thinking about multiple models

Consider several models

$$\mathcal{M}_k : y = f_k(\theta) + \varepsilon_k, k = 1, \ldots, K$$

Then really our posterior is given this

$$p(\theta|y, \mathcal{M}_k) \propto |W_k + \Sigma_k(\theta)|^{-1/2}$$
$$\exp\left(-\frac{1}{2}(y - \mu_k(\theta))^{\mathsf{T}}(W_k + \Sigma_k(\theta))^{-1}(y - \mu_k(\theta))\right) p(\theta).$$

ε_k—the error of the kth model.

BMA

What if we presume that one model is right, but we just don't know which one?

$$p(\mathcal{M}_k) = \text{the prior probability that model } k \text{ is right.}$$
$$p(\mathcal{M}_k|y) = \text{the posterior probability that model } k \text{ is right.}$$

$$p(\mathcal{M}_k, \theta|y) \propto p(y|\theta, \mathcal{M}_k)\, p(\theta, \mathcal{M}_k).$$

$$p(y^*|y) = \sum_{k=1}^{K} \int_{\Theta} p(y^*, \theta, \mathcal{M}_k|y) \mathrm{d}\theta.$$

A tale of two models

Two models are listed: $f_1(\cdot, \theta), f_2(\cdot, \theta)$

$f_\dagger$: Some ideal model that combines things.

Then our data is

$$y_i \sim \text{Normal}(f_\dagger(x_i, \theta), \sigma^2)$$

Say we have priors

- $f_\dagger(x_i, \theta) | f_1(x_i, \theta) \sim \text{Normal}(f_1(x_i, \theta), v_1)$
- $f_\dagger(x_i, \theta) | f_2(x_i, \theta) \sim \begin{cases} \text{Normal}(f_2(x_i, \theta), v_2) & \text{if } i = 1 \\ f_2(x_i, \theta) & \text{if } i > 1 \end{cases}$
- $p(\mathcal{M}_1, \theta) = p(\mathcal{M}_2, \theta) = \frac{1}{2} p(\theta)$.

A tale of two models

If we use $v_2 \to \infty$,

$$p_{\text{BMA}}(\theta|y) \propto \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\frac{(y_i - f_1(x_i,\theta))^2}{\sigma^2 + v_1}\right) p(\theta).$$

We threw out model 2 using BMA!

Model 2 was perfect besides one point, because that point was arbitrarily bad, it breaks.

A tale of two models

Consider the alternative mixing prior, where we are not presuming a model is **right**.

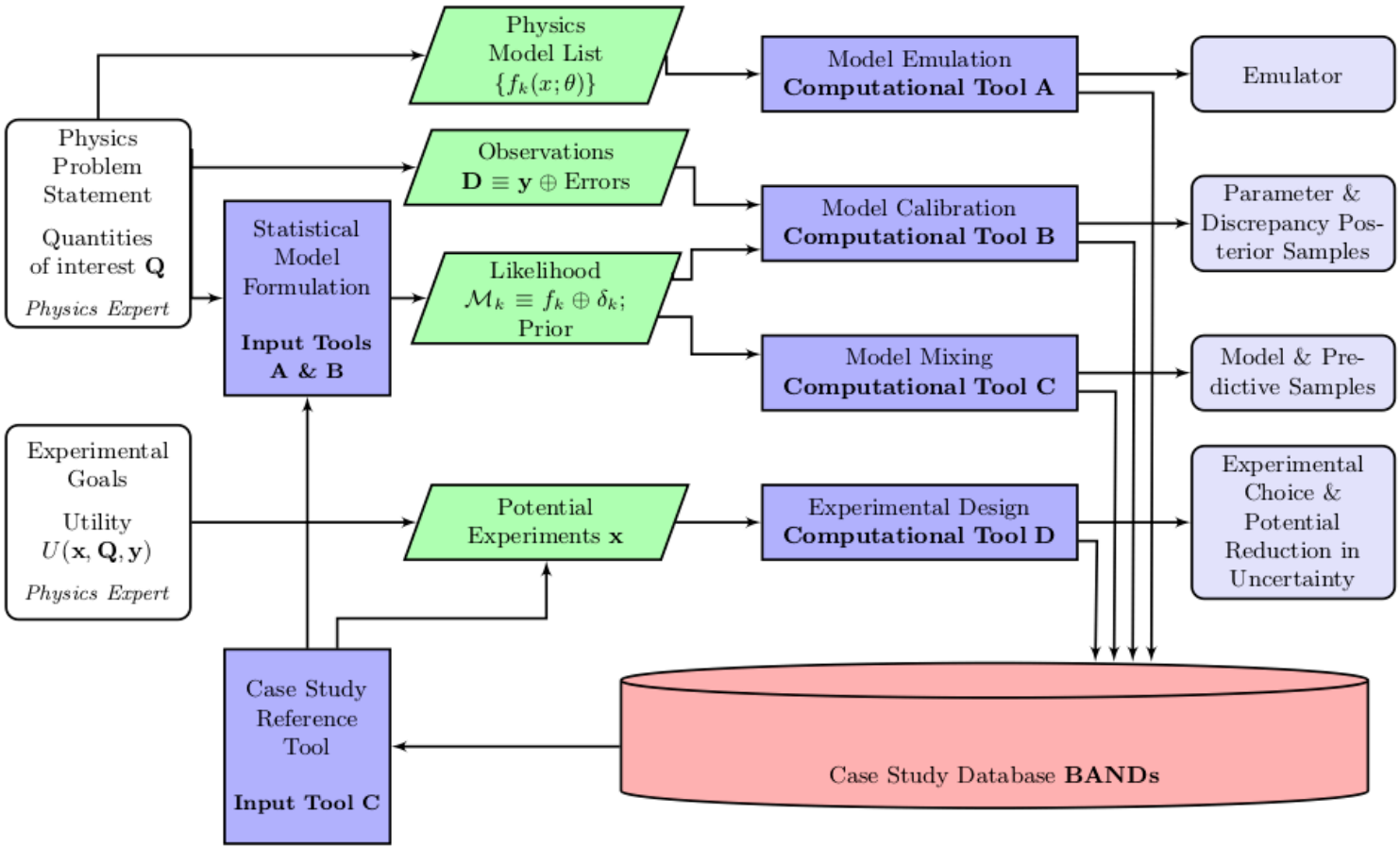
$$f_\dagger(x_i,\theta) \text{ is distributed as } \begin{cases} \mathcal{N}\left(\frac{v_2 f_1(x_i,\theta)+v_1 f_2(x_i,\theta)}{v_1+v_2}, \frac{v_1 v_2}{v_1+v_2}\right) & \text{if } i=1 \\ f_2(x_i,\theta) & \text{if } i>1 \end{cases}$$

Then as $v_2 \to \infty$

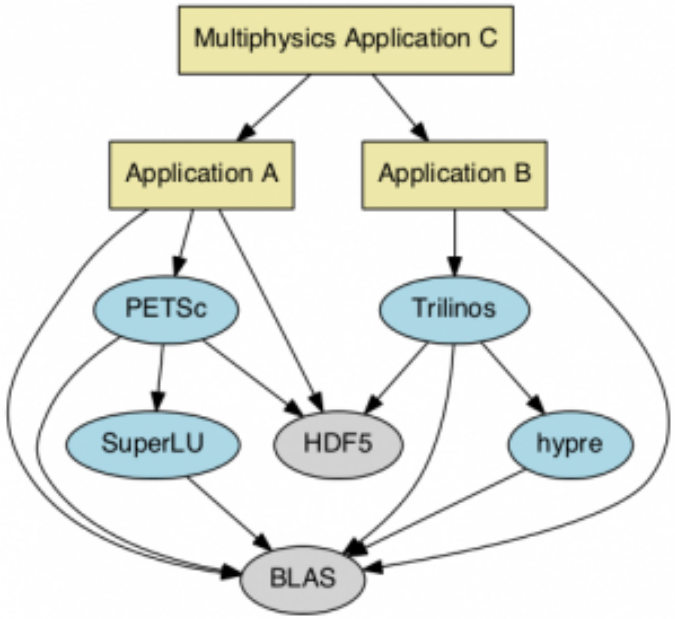
$$p_{\text{BMM}}(\theta|y) \propto \exp\left(-\frac{1}{2}\frac{(y_1 - f_1(x_1,\theta))^2}{\sigma_1^2 + v_1} - \frac{1}{2}\sum_{i=2}^{n}\frac{(y_i - f_2(x_i,\theta))^2}{\sigma^2}\right) p(\theta).$$

No longer threw out model 2!
BAND will focus on enabling these type of more complicated mixing cases.

The BAND framework



SDK (or Software Development Kit)

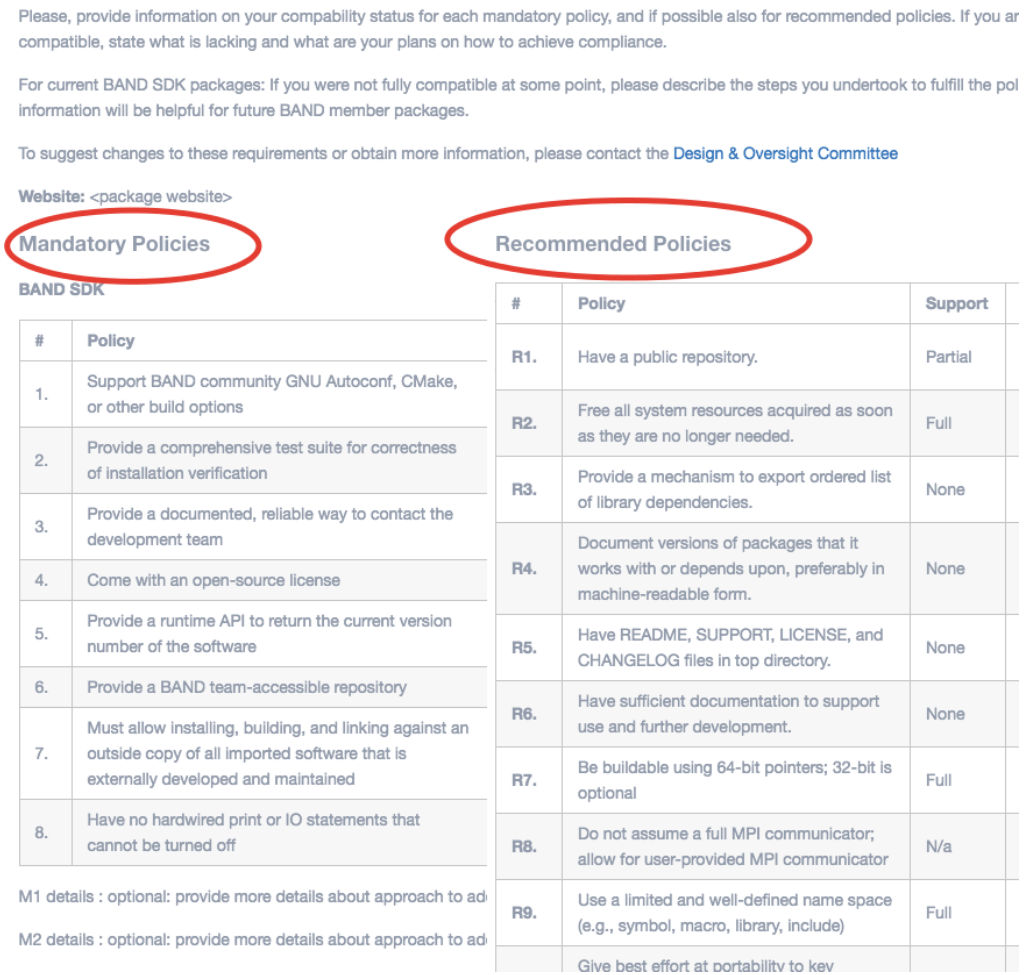


- **Difficulties in combined use of independently developed software packages**
  - Need for consistency among packages to be used within the same application in terms of compiler, compiler versions and options, and common third-party packages such as BLAS and HDF5
  - Namespace conflicts
  - Incompatible versioning

[1] Figure is taken from http://xsdk.info/example-usage/

BAND SDK community policies

To help developers of packages who are considering compatibility with BAND SDK community policies, we provide:



Please, provide information on your compability status for each mandatory policy, and if possible also for recommended policies. If you ar compatible, state what is lacking and what are your plans on how to achieve compliance.

For current BAND SDK packages: If you were not fully compatible at some point, please describe the steps you undertook to fulfill the pol information will be helpful for future BAND member packages.

To suggest changes to these requirements or obtain more information, please contact the Design & Oversight Committee

Website: <package website>

Mandatory Policies

BAND SDK

#	Policy
1.	Support BAND community GNU Autoconf, CMake, or other build options
2.	Provide a comprehensive test suite for correctness of installation verification
3.	Provide a documented, reliable way to contact the development team
4.	Come with an open-source license
5.	Provide a runtime API to return the current version number of the software
6.	Provide a BAND team-accessible repository
7.	Must allow installing, building, and linking against an outside copy of all imported software that is externally developed and maintained
8.	Have no hardwired print or IO statements that cannot be turned off

M1 details : optional: provide more details about approach to ad

M2 details : optional: provide more details about approach to ad

Recommended Policies

#	Policy	Support
R1.	Have a public repository.	Partial
R2.	Free all system resources acquired as soon as they are no longer needed.	Full
R3.	Provide a mechanism to export ordered list of library dependencies.	None
R4.	Document versions of packages that it works with or depends upon, preferably in machine-readable form.	None
R5.	Have README, SUPPORT, LICENSE, and CHANGELOG files in top directory.	None
R6.	Have sufficient documentation to support use and further development.	None
R7.	Be buildable using 64-bit pointers; 32-bit is optional	Full
R8.	Do not assume a full MPI communicator; allow for user-provided MPI communicator	N/a
R9.	Use a limited and well-defined name space (e.g., symbol, macro, library, include)	Full
	Give best effort at portability to key	

https://bandframework.github.io/software/

Example: `surmise`

`surmise` is one element of the BAND collaboration.

`surmise` it is designed to better interface surrogates (or emulators) with calibration/mixing/design tools.

Current tools

- `emulation`: Carries out Bayesian emulation of computer model output and generates inputs to calibration
- `calibration`: Generates estimates of the calibration parameters based on field observations of the real process and an output from emulation
- `utilities`: Performs different utility tasks such as a sampler (e.g., Metropolis-Hastings) to generate posterior draws of calibration parameter

Example: surmise

-Use an external package:

```
    emulator(x=x, theta=theta, f=f, method='GPy', args=args)
```

-Use an existing method to benchmark:

```
    calibrator(emu=emu, y=y, x=x, thetaprior=thetaprior,
                        method='directbayeswoodbury', args=args)
```

-Develop your own method and include into surmise's framework:

```
    calibrator(emu=emu, y=y, x=x, thetaprior=thetaprior,
                        method='mlbayes', args={'clf':clf})
```

Important point: the BAND SDK is designed for people to upload or include difficult-to-run physics code/data problems!

- BAND holds no *license* over the code to be included (if you include your OSS license).
- The collaboration is seeking hard problems that need modern statistical tools of emulation/calibration/mixing.

BAND **not** seek to to develop some golden set of statistical rules but to

- Design useful tools to solve hard NP problems.
- Interface those tools in a stable but flexible way.
- Collocate software infrastructure and interface tools.
- Give documentation and outreach to keep these up to date.