Bayesian Workflow with probabilistic programming

Bob Carpenter
Center for Computational Mathematics
Flatiron Institute
New York

Netherlands eScience SIG, October 2024

Textbook form of workflow

1. Set up a **full probability model**—a joint distribution for observables and unobservables consistent with knowledge about the scientific problem and data collection.

2. **Condition on observed data**: calculate and interpret the **posterior distribution**.

3. **Evaluate the fit and implications**: does it fit data, are conclusions reasonable, is it sensitive to assumptions?

4. **Iterate**: If model fails evaluation, go back to (1).

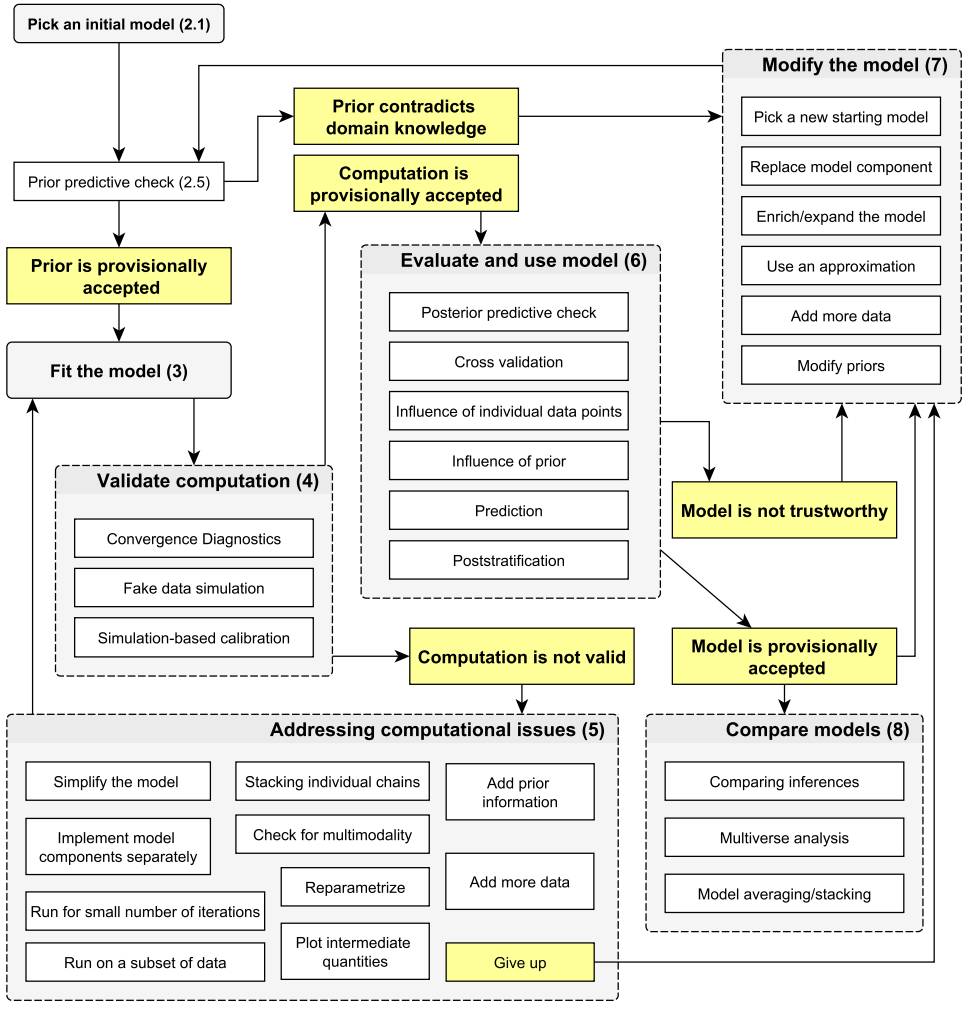
    - For statisticians, last two steps were **revolutionary**.

Gelman et al. 2013. *Bayesian Data Analysis, 3rd Edition*. Chapman & Hall.

Bayesian workflow involves

- **designing experiments** and **collecting data**,
- **designing / coding** probability models,
- evaluating **likelihoods** and **priors**,
- **fitting** models to data,
- **validating** computation,
- addressing **computational issues**,
- **evaluating** and **comparing** model fit and predictions,
- **modifying / improving** models, and
- **deploying** models predictively (prospective or retrospective).

A workflow chart



Gelman et al. 2020. Bayesian worflow. *arXiv*.

Bayesian models

- y is **observed data**, θ are **unknown parameters**
  - suppress unmodeled predictors/features x
- **prior** distribution: $p(\theta)$
- **sampling** distribution: $p(y \mid \theta)$
  - **likelihood** function: $\mathcal{L}(\theta) = p(y \mid \theta)$ for fixed y
- **joint** distribution: $p(y, \theta) = p(y \mid \theta) \cdot p(\theta)$
- **posterior** distribution:

$$p(\theta \mid y) = \frac{p(y, \theta)}{p(y)} = \frac{p(y \mid \theta) \cdot p(\theta)}{\int p(y \mid \theta) \cdot p(\theta) \mathrm{d}\theta} \propto p(y \mid \theta) \cdot p(\theta)$$

Prior and likelihood

- The prior and likelihood can **only be understood together**
  - choosing both is **subjective**, but
  - **likelihood more important**
  - e.g., linear in log odds, compartment ode model, ...
- The prior represents **what you already know**
  - often just **weakly informative** to **determine scale**
- **Sampling distribution** encodes **data generating process**
  - typically a scientific **forward model**
  - coupled with a **noisy observation model**

Estimating gravity (Galielo, Newton 17th c.)

- Roll balls down a ramp and **measure position vs. time** t.
- Solve **Newton's differential equation** for expected position $\hat{y}(t)$ given (a) initial position, (b) slope of ramp, (c) **unknown gravitational constant** G, (d) measurement position.
- Express **prior *knowledge*** as a distribution over plausible values of the gravitational constant: $G \sim \text{normal}(6.7, 0.2)$
- Model the **measurement error** of your process probabilistically $y \sim \text{normal}(\hat{y}, \sigma)$.
- Evaluate the **posterior** distribution to see what you **learned from data**: $p(G \mid y)$.

Science is so cool!

Galileo Museum

What makes inference Bayesian?

- It's **not** the use of a prior.
- It's **averaging over posterior uncertainty**.
- Bayesian inference involves **posterior expectations**, which are defined as posterior averages.

Parameter estimation

- An **estimator** maps data y to estimated parameter value.
- **Bayesian parameter estimate** is posterior expectation

$$\hat{\theta} = \mathbb{E}[\theta \mid y].$$

- The posterior expectation is the estimate that **minimizes expected square error** (over random data sets),

$$\hat{\theta} = \text{argmin}_{\theta} \; \mathbb{E}_{p(y)} \left[\left(\theta - \mathbb{E}\left[\theta \mid y\right] \right)^2 \right].$$

- **Variance** estimates involve an expectation of θ^2.

Event probability

- An **event** C is a subset of parameter space.
- Bayesian **event probability estimate**

$$\Pr[C \mid y] = \mathbb{E}[I_C(\theta) \mid y],$$

  where $\mathrm{I}_C(\theta) = 1$ if $\theta \in C$ and 0 otherwise.

- Events **can be anything**,
  - e.g., $\Pr[\theta > 0.5 \mid y]$, where θ is proportion of male live births, for the event probability of there being more male live births than female (Laplace's original inference problem), with y being records of male births out of total births for 50 years

Posterior predictive distribution

- **Predicts new data** given observed data (and covariates).
- **Posterior predictive distribution**

$$p(\tilde{y} \mid y) = \mathbb{E}[p(\tilde{y} \mid \theta) \mid y],$$

  where $\tilde{y}$ is **new data**, y is **observed data**

- **Averages** prediction of $\tilde{y}$ over uncertainty in θ given y
- Can **condition** everything on covariates (e.g., blood pressure, soil carbon concentration, distance from the moon, etc.)

Expectations via Monte Carlo

- calculate **asymptotically exact** expectations by averaging

$$\begin{aligned}\mathbb{E}[f(\theta) \mid y] &= \int_{\Theta} f(\theta) \cdot p(\theta \mid y) \, \mathrm{d}\theta \\ &= \lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} f(\theta^{(m)}) \\ &\approx \frac{1}{M} \sum_{m=1}^{M} (\theta^{(m)}),\end{aligned}$$

- MCMC **central limit theorem** says that if draws

$$\theta^{(1)}, \ldots, \theta^{(M)} \sim p(\theta \mid y)$$

have **effective sample size** M_{eff}, then **standard error** is

$$\text{se}(\hat{\theta}) = \frac{\text{sd}[\theta \mid y]}{\sqrt{M_{\text{eff}}}}$$

PPLs

- **Probabilistic programming languages** (PPLs)
  - **code** Bayesian **joint densities** (up to constant), and
  - **sample** $\theta^{(n)} \sim p(\theta \mid y)$ from the posterior to compute expectations via Monte Carlo

- It turns out that
  we **need more than posterior sampling** for workflow.

Prior predictive checks

- Prior predictive checks **simulate data** from the marginal

$$y^{\text{sim}} \sim p(y)$$

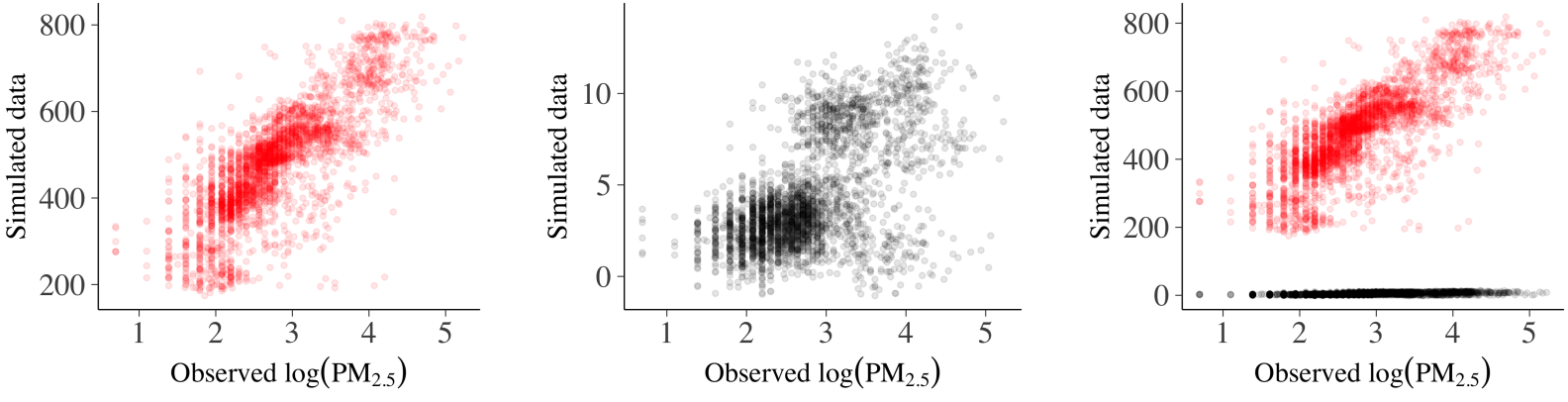
- often by simulating from the **joint**, by generating from **prior** and **sampling** distributions

$$\theta^{\text{sim}} \sim p(\theta) \qquad y^{\text{sim}} \sim p(y \mid \theta^{\text{sim}})$$

- Then **compare** simulated data y^{sim} to observed y

Gabry, Simpson, Vehtari, Betancourt, Gelman. 2019. Visualization in Bayesian workflow. *JRSS A*.

Prior predictive example



(a) Vague priors (b) Weakly informative priors (c) Comparison

- **particulate matter pollution** model with prior on $\log(\mathrm{PM}_{2.5})$
- **vague prior** generates values as dense as neutron star
- **weakly informative** prior controls scale
- subtle with priors on **interacting parameters**
  - why we need a PPL!

How does Stan fare?

- Stan model for **posterior inference**

```
data { int<lower=0> N; int<lower=0, upper=1> y[N]; }
parameters { real<lower=0, upper=1> theta; }
model { theta ~ beta(2, 10);  y ~ binomial(theta); }
```

- Simulate $\theta^{\text{sim}} \sim p(\theta)$ with $N = 0$, but **can't simulate** y!
- Need **alternative Stan model** for **prior predictive checks**

```
data { int N; }
parameters { real<lower=0, upper=1> theta; }
model { theta ~ beta(2, 10); }
generated quantities {
  int y_sim[N] = bernoulli_rng(N, theta);
}
```

How do other PPLs fare?

- **PyMC** typically declares data but **doesn't have to**:

```
y_obs = pm.Normal("y_obs", mu=X @ weights,
                  sigma=noise, observed=y)
```

- **ADMB** declares data in a **DATA_SECTION**
- **Pyro** uses effect handler **condition()** for data, e.g.,

```
poutine.condition(model, data={"z": 1.0})
```

- **Turing.jl** assigns data variables before just-in-time compilation; values may be specified **missing**
- **BUGS** sets data at run time w.r.t. its neutral graphical model

```
theta ~ dbeta(2, 10); for (n in 1:N) y[n] ~ dbern(theta);
```

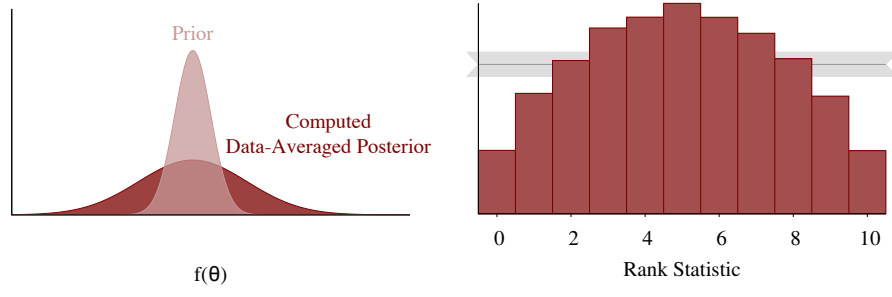
Simulation-based Calibration

- to **validate inference** w.r.t. well-specified data
  - approximate inference like VI will fail
- draw $\theta^{\text{sim}} \sim p(\theta)$ from the prior
- draw $y^{\text{sim}} \sim p(y \mid \theta^{\text{sim}})$ from the sampling distribution
- draw $\theta^{(1)}, \ldots, \theta^{(M)} \sim p(\theta \mid y^{\text{sim}})$ from algorithm to test
- because $(y^{\text{sim}}, \theta^{\text{sim}}) \sim p(y, \theta)$ and $(y^{\text{sim}}, \theta^{(m)}) \sim p(y, \theta)$, θ^{sim} and the $\theta^{(m)}$ should be **identically distributed**
- Allows a **statistical test** of uniformity w. marginal ranks.

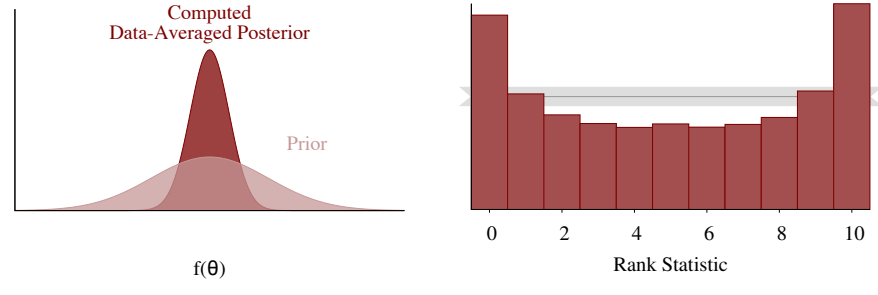
Cook, Gelman, Rubin. 2006. Validation of software for Bayesian models using posterior quantiles. *JCGS*.

SBC diagnoses

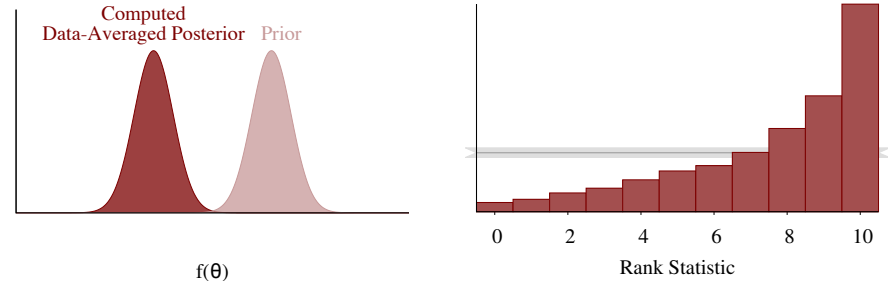
- **over-dispersed**:



- **under-dispersed**:



- **skewed**:



How do PPLs fare on SBC?

- simulation-based calibration requires simulating from prior and sampling distribution
- presents same problem with data specification as prior predictive checks

Posterior predictive checks

- Simulate new data from posterior for draws $m \in 1{:}M$,

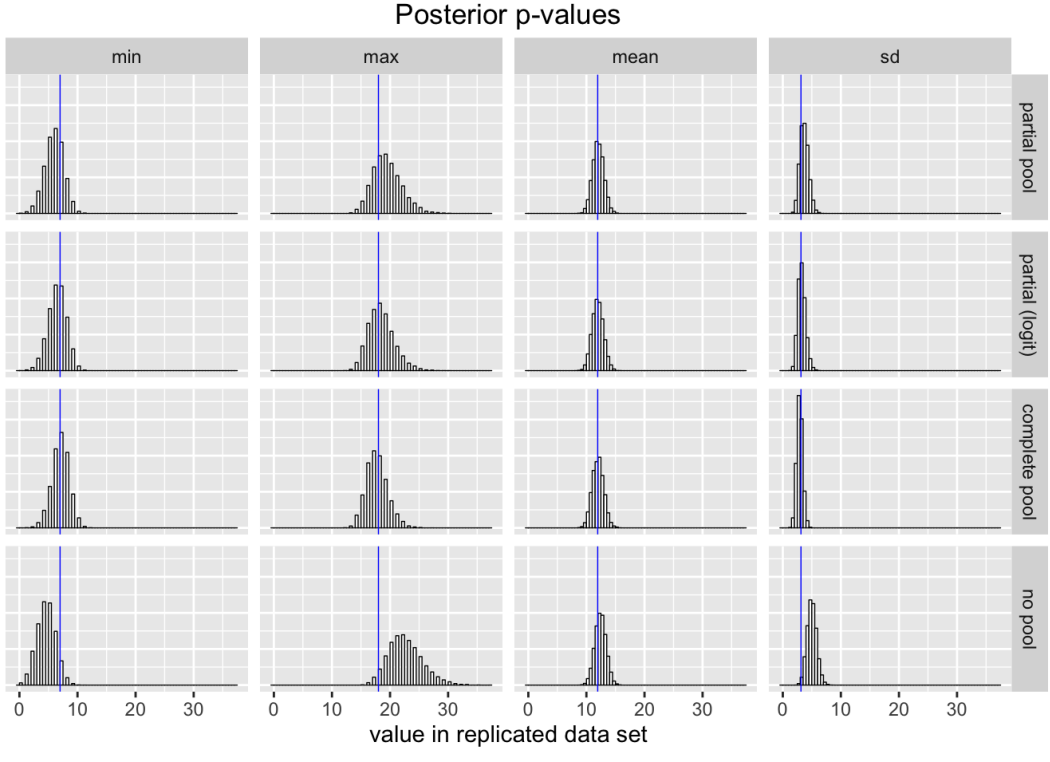
$$\begin{aligned} \theta^{(m)} &\sim p(\theta \mid y) \\ y^{\text{sim}(m)} &\sim p(y \mid \theta^{(m)}) \end{aligned}$$

- Compare statistics $s(y)$ on observed data to those of posterior simulations $s(y^{\text{sim}(m)})$, e.g.,
  - $s()$ can be anything, e.g., mean, max, sd, quantiles, ranks, skew, etc.

- Plot, or compute two-sided posterior p-values to automate,

$$\begin{aligned} p\text{-value} = \min(\ & \Pr[s(y) < s(y^{\text{sim}})], \\ & 1 - \Pr[s(y) < s(y^{\text{sim}})] \ ) \end{aligned}$$

Posterior predictive example



- model of repeated binary trials (baseball batting avg.)
  - vertical line is $s(y)$, histogram is $s(y^{\text{sim}(m)})$
  - max() and sd() statistics "reject" the no-pooling model

PPL support for PPCs

- requires extracting posterior draws and simulating data from them
- still the same problem of flexibly specifying data vs. parameters (i.e., knowns vs. unknowns)

Cross-validation

- divide data into train/test split (say y and $\tilde{y}$)
- fit model on training set
- evaluate predictive log density on test set,

$$\begin{aligned} \log p(\tilde{y} \mid y) &\approx \log \frac{1}{M} \sum_{m=1}^{M} p(\tilde{y} \mid \theta^{\text{sim}(m)}) \\ &= \text{log-sum-exp}_{m=1}^{M} \log p(\tilde{y} \mid \theta^{\text{sim}(m)}) - \log(M) \end{aligned}$$

PPL support for X-val

- fit with one data set y and evaluate with another $\tilde{y}$
- **BUGS** almost succeeds

```
for (n in 1:N) y[n] ~ dnorm(alpha + x[n] * beta, tau)
tau ~ gamma(1, 1);  alpha ~ normal(0, 2);  beta ~ normal(0, 2)
```

  by letting $y = y^{\text{train}}, y^{\text{test}}$ be partially missing
  - but doesn't let you retrieve the log density values for y^{test}
  - this also seamlessly handles missing data (that's modeled)
- **Turing.jl** allows the same thing (values?)
- other PPLs require additional sampling statements for the test data

Stan for X-val

- Stan codes **leave-one-out X-val** by specifying test point

```
data {
  int N; int[N] y;  int nt;
}
parameters {
  real mu; real<lower=0> sigma;
}
model {
  append_row(y[:nt-1], y[nt+1:]) ~ normal(mu, sigma);
  mu ~ normal(0, 1);  sigma ~ lognormal(0, 1);
}
generated quantities {
  real lp = normal_lpdf(y[nt] | mu, sigma);
}
```

- but it's a **totally different model**

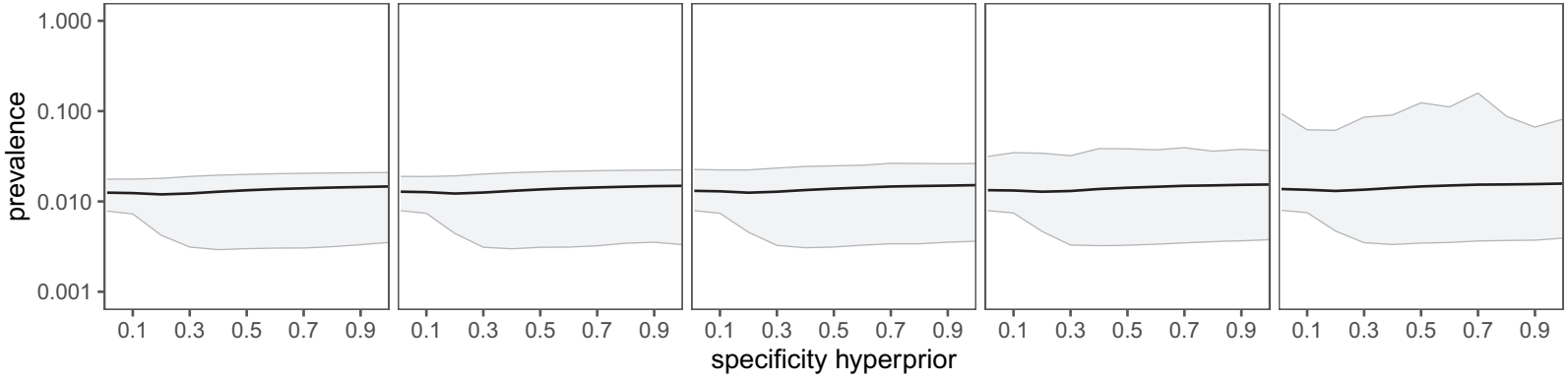
Sensitivity analysis

- we'd like to understand how **changes in our model affect posterior inference**
- e.g., vary priors and see how posterior expectations changes
- all PPLs let you evaluate alternative constants easily
- **derivative-based sensitivity** w.r.t. const. c is trickier

$$\frac{\partial}{\partial c}\mathbb{E}[f(\theta) \mid y, c]$$

Ryan Giordano modified Stan's C++ to compute this for his (Berkeley) Ph.D. thesis, but it's not exposed

Sensitivity example



- Estimated Covid seroprevalence (y axis) as a function of
  - the hyperprior for specificity (x-axis)
  - the hyperprior for sensitivity (facets with values from left-to-right 0.01, 0.25, 0.5, 0.75, 1.0)

- Gelman, Carpenter. 2020. Bayesian analysis of tests with unknown specificity and sensitivity. *JRSS C*.

Workflow goes beyond inference

- **clamp/pin parameters** to fixed values?
  - Stan requires moving the variable from the parameter to the data block
- working with **multiple (related) models**?
  - model comparison
  - model reparameterization
  - model averaging/mixing/stacking
- **autogenerating** concurrent or GPU code?
  - Stan requires using parallel map functions in the program

‘

Naming and persistence is hard

- how to **name and store multiple model variants**?
  - `uk-covid-icar`, `uk-covid-rw1`, `uk-covid-rw2`, `uk-covid-rw2-icar`, `uk-covid-rw2-bym2`, `uk-covid-rw2-bym2pc`, `uk-covid-rw2-bym2pc-no-socio`, *ad infinitum* ...
  - plus multiple versions of the same model (over time)
- how to **name and store output**?
- how to work with **distributed teams**?
  - e.g., how to **share results** given that samples can be large?
  - or that they run on clusters in pieces

Other workflow issues

- data may be tied up with **privacy** and/or **intellectual property** concerns
  - e.g., medical records, search logs, street views, etc.
- end application may require **deployment in production**
  - bundle with Docker, or otherwise deploy
  - robustness is a key issue
  - update as new data comes in

- **What are we missing?**

Democratization of modeling

- **expression-based iterfaces** use PPLs under the hood, but give users simpler specification sublanguages
  - **brms**: expression interface in R
  - a Poisson GLM with log link is a one-liner

```
y ~ age + base * treatment + (1 | patient)
```

- **fully encapsulated interfaces** use PPLs under the hood but give users a menu of model choices
  - **Prophet** (time-series with trends and cycles)
  - **Torsten** (PK/PD compartment models)
- these systems involve **lots of defaults**
  - **evaluation crosses application boundaries**

Elephant in the room: Modularity

- how to **modularize model components** like hierarchical priors or GP priors?
- Stan lets users define **functions**
  - e.g., a random-walk or ICAR prior's density function
- but they **can't cross block boundaries**, e.g., data, parameter, model, generated quantities
- what about other PPLs?
- a residual problem: **density is modular, behavior isn't**
  - e.g., a prior can only be understood in the context of a likelihood and a data set

References

- **workflow paper**
  - Gelman, Vehtari, Simpson, Margossian, Carpenter, Yao, Kennedy, Gabry, Bürkner, Modrák. 2020. Bayesian workflow. *arXiv*.
- open-access **workflow book**
  - Above authors give or take. 2025(?) *Bayesian Workflow*. Chapman & Hall/CRC.
  - GitHub repo: `https://github.com/jgabry/bayes-workflow-book`