ArviZ: a unified library for Bayesian model criticism and visualization in Python

Colin Carroll
Freebird, Inc.
colin@freebird.com

Austin Rochford
Monetate, Inc.
arochford@monetate.com

1 Data Structure

Bayesian inference produces naturally high dimensional data: in the case of Markov chain Monte Carlo, it is common to produce multiple independent simulations (chains) to facilitate calculations like effective sample size and Gelman-Rubin statistics. In this case, posterior samples are of dimension at least 2, and higher for multivariate random variables. Storing the data as an **xarray** dataset allows for labeled querying of this data, along with serialization, and attached metadata.

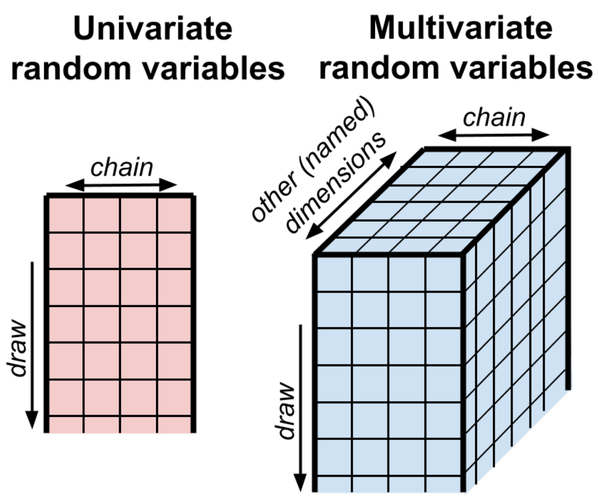

Figure 1: The shape of the xarray objects holding posterior samples for univariate and multivariate random variables

ArviZ stores these multiple datasets from inference using **netCDF** groups, which are themselves built with HDF5. The **InferenceData** class implements this functionality. By using netCDF, in addition to native handling of high dimensional data and being able to use existing serialization and deserialization function, all functions need be implemented only once.

For example, here is how to load data from running 4 chains of NUTS on the radon model of Gelman and Hill using PyMC3.

```
radon = az.load_data('datasets/radon.nc')
Inference data with groups:
        > posterior
        > sample_stats
        > posterior_predictive
        > prior
        > observed_data
```

The following plot illustrates ArviZ's ability to work with posterior inferences from many different probabilistic programming libraries.

2 Plots and Diagnostics

In addition to common plots for Bayesian analysis like trace plots and forest plots, the library implements many of the visualizations from [1], including ppc plots, pair plots, and parallel coordinate plots. Additionally, it supports a number of statistical checks, such as calculating the effective sample size, the Gelman-Rubin statistic, Pareto-smoothed importance sampling leave-one-out cross validation (PSIS-LOO-CV), and widely applicable information criterion (WAIC).

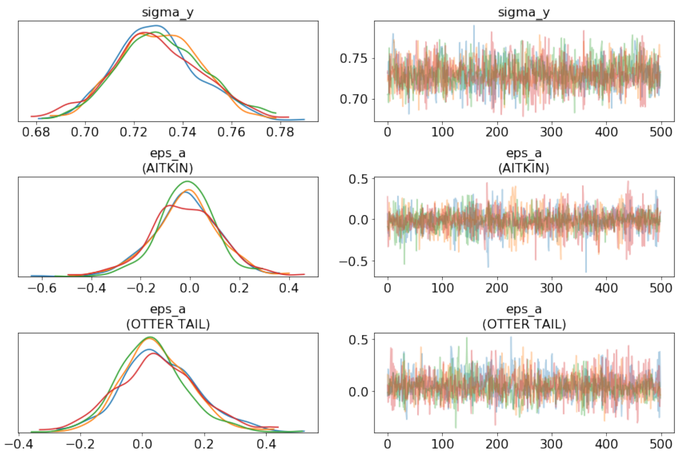

Figure 2: A traceplot in ArviZ from the radon example, where the user specified the variable `eps_a` and the counties `AITKIN` and `OTTER TAIL`.

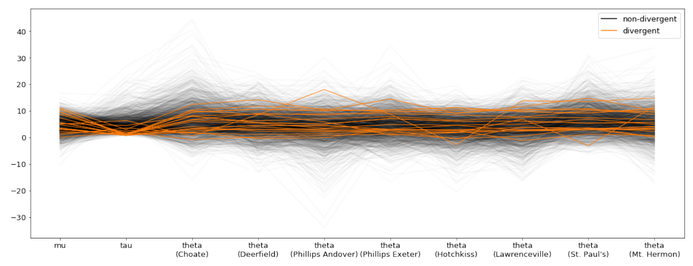

Figure 3: A parallel coordinate plot in ArviZ from the eight schools example, highlighting divergences when tau is small.

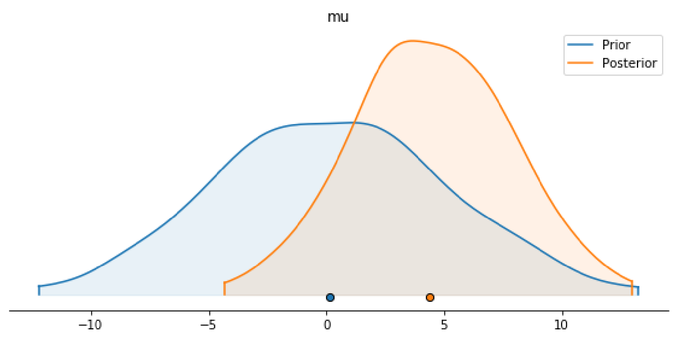

Figure 4: Compoaring the prior and the posterior of a parameter in ArviZ from the eight schools example.

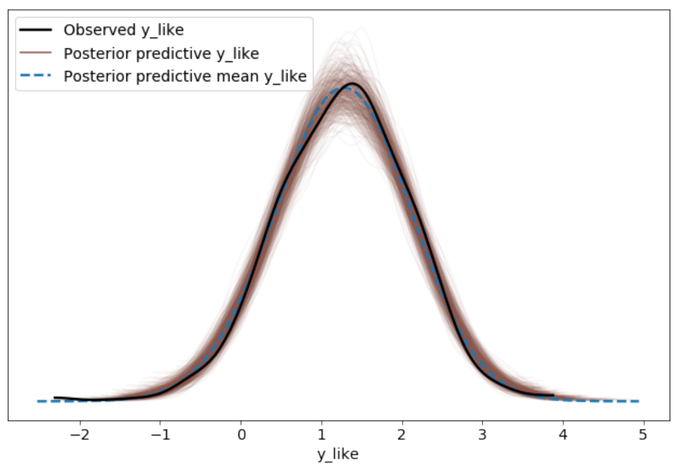

Figure 5: A posterior predictive plot in ArviZ from a Minnesota radon model.

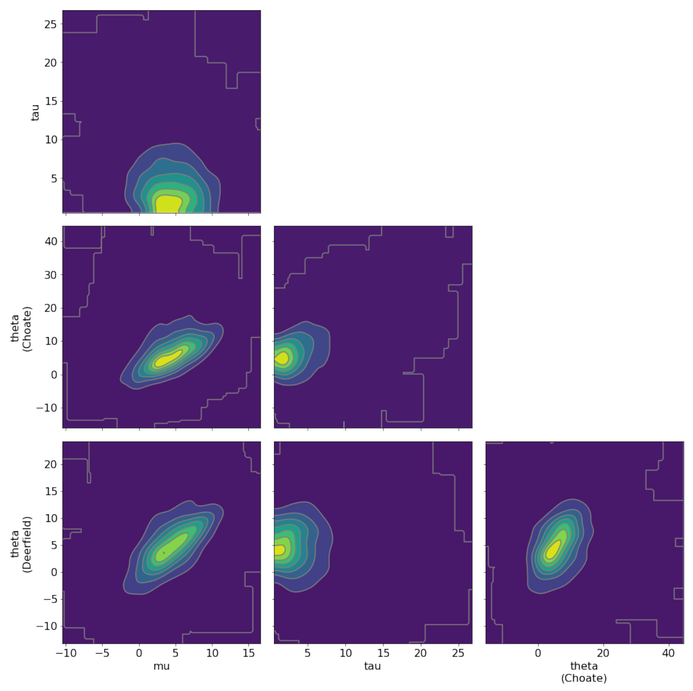

Figure 6: A pair plot plot in ArviZ from the eight schools example.

school	mu	theta	tau
Choate	199.94	452.30	213.8
Deerfield	199.94	514.82	213.8
Phillips Andover	199.94	500.70	213.8
Phillips Exeter	199.94	481.20	213.8
Hotchkiss	199.94	397.28	213.8
Lawrenceville	199.94	456.43	213.8
St. Paul's	199.94	537.28	213.8
Mt. Hermon	199.94	561.03	213.8

Table 1: Calculating effective samples for a centered eight schools model using ArviZ. Note that the natural output is in an xarray dataset, which may be flattened into the table above.

3 Interfaces

Package	Interface	Sampler stats	Posterior predictive
PyMC3	Native	Yes	Yes
PyStan, CmdStan	Native	Yes	Yes
Pyro	Native	No	No
Emcee	Native	No	No
Edward	Dict	No	No

As an explicit example, one may use PyMC3 to sample from the prior predictive, posterior, and posterior predictive distributions, and then convert that into an InferenceData object:

```
with model:
    prior = pm.sample_prior_predictive()
    trace = pm.sample()
    ppc = pm.sample_posterior_predictive(trace)

radon = arviz.from_pymc3(
    trace=trace,
    prior=prior,
    posterior_predictive=ppc,
    coords={
        'county': mn_counties,
        'observed_county': observed_counties
    },
    dims={
        'eps_a': ['county'],
        'a': ['observed_county'],
        'mu_a': ['observed_county'],
        'y_like': ['observed_county']
    })
```

This creates the radon data from the first section. While we specified certain parts of the data, note that ArviZ also used the available objects to extract observed data, which is an attribute on the trace, and sample stats, which records metrics like energy, tree depth, and whether there was a divergence. These are indexed using the same "draw" and "chain" dimensions that the other parameters use, simplifying queries.

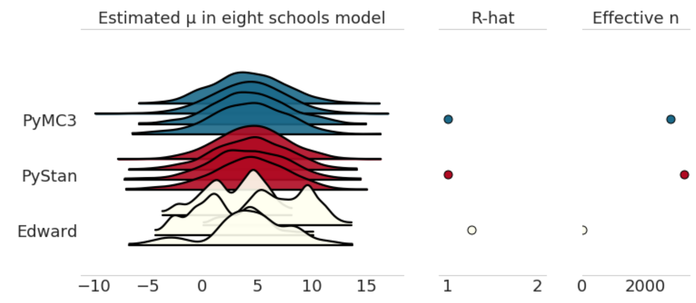

Figure 7: An example ridgeplot in ArviZ comparing estimates from the same model in three libraries, suggesting a poorly tuned sampler from Edward

4 Acknowledgements

The ArviZ project is open source, but has seen significant contributions from Osvaldo Martin, Ari Hartikainen, Ravin Kumar, and Adrian Seyboldt.

5 References

http://arviz-devs.github.io/arviz

The following QR code links to a Jupyter notebook containing the ArviZ code necessary to reproduce the plots in this poster.

[1] Jonah Gabry, Daniel Simpson, Aki Vehtari, Michael Betancourt, and Andrew Gelman. Visualization in bayesian workflow. *arXiv preprint arXiv:1709.01449*, 2017.