ArviZ a unified library for exploratory analysis of Bayesian models in Python

Ravin Kumar\textsuperscript{1}, Colin Carroll\textsuperscript{1}, Ari Hartikainen\textsuperscript{2}, and Osvaldo Martin\textsuperscript{3}

\textsuperscript{1}Freebird Inc., United States \textsuperscript{2}Aalto University, Department of Civil Engineering, Espoo, Finland \textsuperscript{3}Instituto de Matemática Aplicada San Luis, UNSL-CONICET. Ejército de los Andes 950, 5700 San Luis, Argentina \textsuperscript{4}Carbon IT LLC, United States

Summary

While conceptually simple, Bayesian methods can be mathematically and numerically challenging. Probabilistic programming languages (PPLs) implement functions to easily build Bayesian models together with efficient automatic inference methods. This helps separate the model building from the inference, allowing practitioners to focus on their specific problems and leaving PPLs to handle the computational details for them (Bessiere, Mazer, Ahuactzin, & Mekhnacha, 2013; Daniel Roy, 2015; Ghahramani, 2015). The inference process generates a posterior distribution — which has a central role in Bayesian statistics — together with other distributions like the posterior predictive distribution and the prior predictive distribution. The correct visualization, analysis, and interpretation of these distributions is key to properly answer the questions that motivate the inference process.

When working with Bayesian models there are a series of related tasks that need to be addressed besides inference itself:

- Diagnoses of the quality of the inference
- Model criticism, including evaluations of both model assumptions and model predictions
- Comparison of models, including model selection or model averaging
- Preparation of the results for a particular audience

Successfully performing such tasks are central to the iterative and interactive modeling process. These tasks require both numerical and visual summaries to help statisticians or practitioners analyze visual summaries. In the words of Persi Diaconis (Diaconis, 2011) “Exploratory data analysis seeks to reveal structure, or simple descriptions in data. We look at numbers or graphs and try to find patterns. We pursue leads suggested by background information, imagination, patterns perceived, and experience with other data analyses”.

For these reasons we introduce ArviZ, a Python package for exploratory analysis of Bayesian models. ArviZ aims to be a package that integrates seamlessly with established probabilistic programming languages like PyStan (“Stan,” n.d.), PyMC (Salvatier, Wiecki, & Fonnesbeck, 2016), Edward (Tran et al., 2017, 2016), emcee (Foreman-Mackey, Hogg, Lang, & Goodman, 2013), Pyro (Bingham et al., 2018), and easily integrated with novel or bespoke Bayesian analyses. Where the aim of the probabilistic programming languages is to make it easy to build and solve Bayesian models, the aim of the ArviZ library is to make it easy to process and analyze the results from the Bayesian models. We hope ArviZ will become a key Python tool for Bayesian data analysis by allowing users to focus on problems from their domain knowledge and not on computational details.

Bayesian inference produces naturally high dimensional data. By storing each type of data in its natural format before storing them in a high dimensional data array, the user can get to the relevant data dimensions and their numerical or visual summaries in a matter of seconds. This also allows for more interactive exploratory analysis and simplifies the process of manipulations and analysis.

data resulting from PPLs as an xarray (Hoyer & Hamman, 2017) dataset, ArviZ provides labeled querying of the data, efficient algorithms, and persistent metadata. These datasets are stored together on disk and in code using netCDF4 (Brown, Folk, Goucher, Rew, & Dubois, 1993; R. Rew & Davis, 1990) groups, which are themselves built with HDF5, and allows for well supported serialization. This functionality is implemented in the InferenceData class (see Figure 1). In addition to the listed benefits of using netCDF and xarray, by using a single data structure all statistical and visualization functions need to be implemented only once.

In addition to common plots for Bayesian analysis including a trace plot and forest plot, ArviZ implements other visualizations such as a plot for posterior predictive checks, a pair plot, and a parallel coordinate plot (Gabry, Simpson, Vehtari, Betancourt, & Gelman, 2017). Additionally, it supports a number of statistical checks, such as calculating the effective sample size, the r-hat statistic, Pareto-smoothed importance sampling leave-one-out cross validation (PSIS-LOO-CV) (Vehtari, Gelman, & Gabry, 2015), and widely applicable information criterion (WAIC) (Watanabe, 2013).

**Funding**

Work by Osvaldo Martin was supported by CONICET-Argentina and ANPCyT-Argentina (PICT-0218).

**Acknowledgments**

We thank the PyMC3 Community — especially Adrian Seyboldt, Jumpeng Lao, and Thomas Wiecki — as well as the Stan community — especially Allen Riddell. We also would like to extend thanks to all the ArviZ contributors, and the contributors of the libraries used to build ArviZ — particularly xarray, matplotlib, pandas, and numpy.

**Example plots**

A portion of ArviZ’s functionality is shown in Figure 2 through Figure 5. ArviZ supports more plotting and inference diagnostics, in addition to the ones shown here. Additional examples can be found ArviZ documentation gallery.
Figure 2: Bivariate hexbin plot with marginal distributions

Figure 3: 2D Kernel Density estimation

Figure 4: Markov Chain Monte Carlo Trace Plot

Figure 5: John Kruschke styled posterior distribution plots
References


