ArviZ: backend agnostic exploratory analysis of Bayesian models in Python

ORIOL ABRIL, Universitat Pompeu Fabra
COLIN CARROLL, ARI HARTIKAINEN, RAVIN KUMAR, and OSVALDO MARTIN

Probabilistic programming is an emerging field of growing importance. In recent years, many libraries have been built to write probabilistic models as executable code. ArviZ proposes the use of a common data structure called `InferenceData` to ease the analysis of the results. To this end, ArviZ converts results from several libraries to `InferenceData`. It also provides functions to plot the results —using matplotlib or bokeh—, calculate relevant diagnostics or perform model checing. We therefore hope ArviZ will become a key tool in Bayesian analysis by providing users with computational tools to analyze and explore their results and to compare between different probabilistic programming libraries.

1 INTRODUCTION

The growth of both probabilistic programming languages and algorithms for the analysis of Bayesian inference results led to the introduction of the Python library ArviZ [Kumar et al. 2019]. Outside inference itself and sampling, ArviZ covers all the other aspects of Bayesian analysis. It aims to ease the use of a robust Bayesian workflow and cutting edge diagnostics so users can concentrate on domain specific knowledge.

Following the structure of ArviZ's source code itself, this paper divides all these tasks into 3 categories, each of which has its own section. Section 2 is about data storage of Bayesian inference results. Our goal is to define a storage schema compatible with netCDF [Unidata 2011]. This data structure should contain all relevant information to perform diagnostics on the inference run or to reproduce it. Sections 3 and 4 outline the algorithms and plotting functions implemented in ArviZ respectively.

2 DATA STORAGE SCHEME

One of ArviZ key features is its data storage scheme using `InferenceData` objects. `InferenceData` (built on top of xarray [Hoyer and Hamman 2017]) contain several groups, each of which is a multidimensional labeled array with one or several data variables (Figure 1).

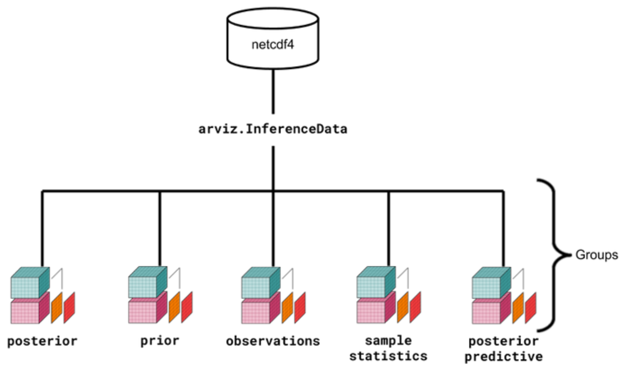


Fig. 1. Visual representation of the `InferenceData` structure.

Authors' addresses: Oriol Abril, Universitat Pompeu Fabra, Barcelona, `oriol.abril.pla@gmail.com`; Colin Carroll; Ari Hartikainen; Ravin Kumar; Osvaldo Martin.

These groups are thought to contain a specific quantity such as *posterior*, *observed data* or *sampler stats* (sampler quantities like divergencies, tree depth, or log likelihood). Groups in `InferenceData` directly translate to groups defined in netCDF files, making the storage of `InferenceData` objects in netCDF files straightforward. The full schema specification can be found on ArviZ documentation[1].

`InferenceData` objects are central to ArviZ, most ArviZ functions take `InferenceData` as input. Therefore, we provide native functions to convert between the outputs of several inference libraries and `InferenceData`. Table 1 shows the state of converter functions in ArviZ beta release 0.6.1 (latest release to date). Natively supported libraries are PyStan, CmdStan and CmdStanPy [Stan Development Team 2018a,b,c,d], PyMC3 [Salvatier et al. 2016], Pyro and NumPyro [Bingham et al. 2018], emcee [Foreman-Mackey et al. 2019, 2013] and TensorFlow Probability [Dillon et al. 2017]. In addition, there is also a function to convert dictionaries to `InferenceData`.

Table 1. ArviZ converter functions summary

Inference Library	Sampler Stats	Posterior predictive	Observed data	Prior
PyStan, CmdStan, CmdStanPy	✓	✓	✓	✓
PyMC3	✓	✓	✓	✓
Pyro	✓[2]	✓	✓[2]	✓
NumPyro	✓	✓	✓	✓
emcee	Limited[3]	Limited[3]	✓	✗
tensorflow probability	Limited[4]	✓	✓	✗

All data currenlty stored in `InferenceData` is related to the Bayesian inference run itself, either directly or indirectly. For example, posterior predictive samples stored are intended for model checking and therefore do not really support posterior predictive samples intended for prediction. We are still working on the `InferenceData` scheme with two main goals in mind, support the storage of predictions, and storage of the arguments used when calling the sampler as well as the kind of sampler. This latter goal would allow to reproduce the sampling given the `InferenceData` object.

3 STATISTICS AND DIAGNOSTICS

ArviZ includes many statistics and diagnostics for Bayesian analysis such as the Gelman-Rubin statistic [Gelman and Rubin 1992] or the widely applicable Information Criterion [Watanabe 2010]. We strive to provide sensible defaults for all algorithms and to keep them up to date with the latest publications. Moreover, Numba [Lam et al. 2015] is used to speed up lengthy calculations.

3.1 Diagnostics

Many convergence assessment functions are available in ArviZ. `az.rhat`, `az.ess` and `az.mcse` implement the Gelman-Rubin statistic, effective sample size and Monte Carlo standard error respectively [Gelman et al. 2013; Gelman and Rubin 1992]. The improvements in Vehtari et al. [2019] have already been added to ArviZ and made the default for all 3 functions. Moreover, all 3 functions take

[1] https://arviz-devs.github.io/arviz/schema/schema.html
[2] For Pyro<1.0.0 there is only partial sampler stats support and no observed data available
[3] emcee's blobs can be used to store sampler stats or even posterior predictive samples, however, blobs can store anything, so it is up to the user to customize them properly
[4] Only log likelihood data is retrieved

as input a method to choose between the available versions of the algorithms. For example, the effective sample size can be calculated for the bulk or for quantiles and with or without rank normalization. `az.bfmi` can be used to calculate Bayesian fraction of missing information [Betancourt 2016]. `az.geweke` implements the z-scores for convergence diagnostics [Geweke 1991].

An `az.summary` function is also provided for convenience to calculate several diagnostics and also statistics such as mean or variance. Its output can be directly printed in a readable manner.

3.2 Model comparison

ArviZ also provides functions for model comparison using the predictive accuracy as metric. Given that ArviZ is intended to analysis of Bayesian inference results and it aims to extend best practices, only the fully Bayesian algorithms have been implemented [Gelman et al. 2014]. `az.waic` computes the widely applicable information criterion [Watanabe 2010] and `az.loo` computes the leave one out cross validation using Pareto smoothed importance sampling [Vehtari et al. 2015, 2017].

`az.compare` can be used to compute the WAIC or LOO value for several models at the same time, effectively comparing them in a single line of code.

3.3 Model checking

One algorithm for model checking, `az.loo_pit`, has also been implemented recently, the leave one out cross validation pareto integrated transform (LOO-PIT) [Gabry et al. 2019]. This has been possible thanks to the versatility of `InferenceData`, which contains all the quantities needed for this calculation: observed data, posterior predictive samples and log likelihood data. It should be noted that the LOO-PIT result is generally analyzed visually in a qualitative way, this can be also done in ArviZ using native plotting functions. In addition, the results from `az.loo` and `az.waic` can also be used for model checking to some extent.

4 DATA VISUALIZATION

Many of the algorithms described in Section 3 have complemetary plots that extend them and ease their interpretability. Some examples are `az.plot_compare`, `az.plot_loo_pit` or `az.plot_ess` which complement the algorithms of the same name, but also `az.plot_khat` to complement `az.loo` or `az.plot_elpd` to complement either `az.waic` or `az.loo`.

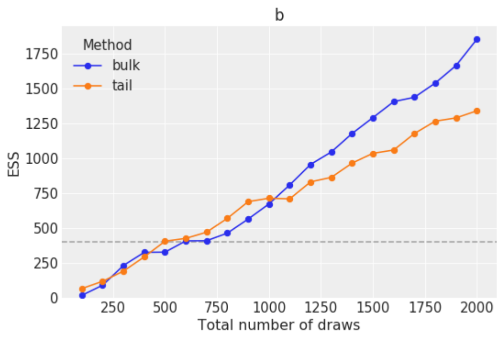
ArviZ provides plotting functions suited for many different tasks: visualization of probabiliy distributions, model checking, convergence diagnostics or model comparison. The documentation contains an example gallery showcasing all plots currently available in ArviZ. Figure 2 shows some of the recently added plots.

FUNDING

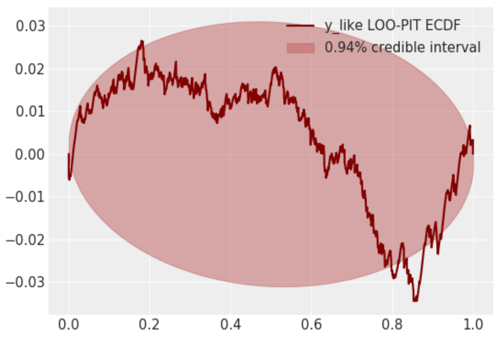
Work by Osvaldo Martin was supported by CONICET-Argentina and ANPCyT-Argentina (PICT-0218). Two students were funded by the Google Summer of Code project to work on ArviZ between May and August 2019.

ACKNOWLEDGMENTS

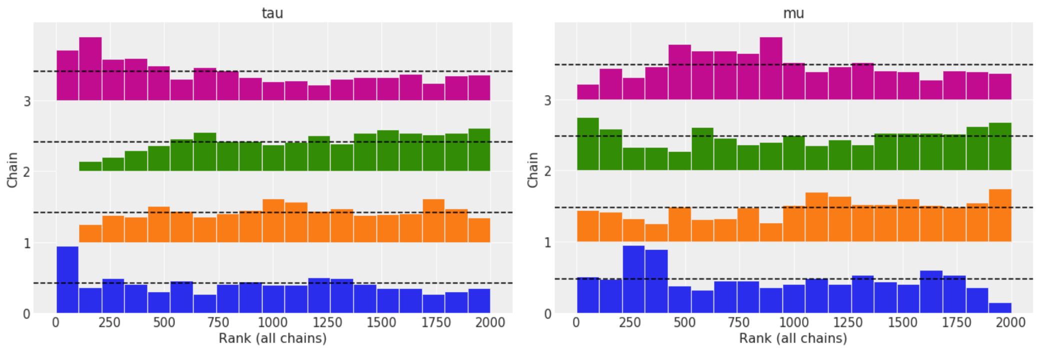
We would like to acknowledge all the comunities involved in ArviZ development —especially Adrian Seyboldt, Junpeng Lao, Thomas Wiecki and Robert Goldman from the PyMC community, Allen Riddell and Aki Vehtari from the Stan community and Du Phan from the Pyro community. We also would like to thank everyone who contributed to ArviZ sending pull requests or posting issues, and the contributors of the libraries used to build ArviZ —particularly xarray, matplotlib, bokeh and numpy.



(a) Evolution of the effective sample size as the sample size increases. A non-linear evolution of the effective sample size indicates convergence problems [Vehtari et al. 2019].



(b) LOO-PIT values estimated probabiliy density function. Differing from a uniform distribution indicates model mispecification [Gabry et al. 2019].



(c) Rank plots are an alternative to the commonly used trace plots [Vehtari et al. 2019].

Fig. 2. Examples of plotting functions added to ArviZ during 2019.

REFERENCES

Michael Betancourt. 2016. Diagnosing suboptimal cotangent disintegrations in Hamiltonian Monte Carlo. *arXiv preprint arXiv:1604.00695* (2016).

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. 2018. Pyro: Deep Universal Probabilistic Programming. *arXiv preprint arXiv:1810.09538* (2018).

Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. 2017. Tensorflow distributions. *arXiv preprint arXiv:1711.10604* (2017).

Daniel Foreman-Mackey, Will M Farr, Manodeep Sinha, Anne M Archibald, David W Hogg, Jeremy S Sanders, Joe Zuntz, Tobias Erhardt Val-Borro, Ilya Pashchenko, and Oriol Abril Pla. 2019. emcee v3: A Python ensemble sampling toolkit for affine-invariant MCMC. *Journal of Open Source Software* (2019). https://doi.org/10.21105/joss.01864

Daniel Foreman-Mackey, David W Hogg, Dustin Lang, and Jonathan Goodman. 2013. emcee: the MCMC hammer. *Publications of the Astronomical Society of the Pacific* 125, 925 (2013), 306.

Jonah Gabry, Daniel Simpson, Aki Vehtari, Michael Betancourt, and Andrew Gelman. 2019. Visualization in Bayesian workflow. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 182, 2 (2019), 389–402.

Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. 2013. *Bayesian data analysis.* Chapman and Hall/CRC, New York, NY, USA. https://doi.org/10.1201/b16018

Andrew Gelman, Jessica Hwang, and Aki Vehtari. 2014. Understanding predictive information criteria for Bayesian models. *Statistics and computing* 24, 6 (2014), 997–1016.

Andrew Gelman and Donald B Rubin. 1992. A single series from the Gibbs sampler provides a false sense of security. *Bayesian statistics* 4 (1992), 625–631.

John Geweke. 1991. *Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments*. Vol. 196. Federal Reserve Bank of Minneapolis, Research Department Minneapolis, MN, Minneapolis.

S. Hoyer and J. Hamman. 2017. xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software* 5, 1 (2017). https://doi.org/10.5334/jors.148

Ravin Kumar, C Colin, Ari Hartikainen, and Osvaldo A Martin. 2019. ArviZ a unified library for exploratory analysis of Bayesian models in Python. *J. Open Source Software* 4 (2019), 1143. https://doi.org/10.21105/joss.01143

Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. 2015. Numba: A LLVM-Based Python JIT Compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC (LLVM '15)*. Association for Computing Machinery, New York, NY, USA, Article Article 7, 6 pages. https://doi.org/10.1145/2833157.2833162

John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. 2016. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science* 2 (2016), e55.

Stan Development Team. 2018a. PyStan: the Python interface to Stan. http://mc-stan.org.

Stan Development Team. 2018b. The Stan Core Library. http://mc-stan.org.

Stan Development Team. 2018c. The Stan Math Library. http://mc-stan.org.

Stan Development Team. 2018d. Stan Modeling Language Users Guide and Reference Manual. http://mc-stan.org.

Unidata. 2011. Network Common Data Form (netCDF). (2011). https://doi.org/10.5065/D6H70CW6

Aki Vehtari, Andrew Gelman, and Jonah Gabry. 2015. Pareto smoothed importance sampling. *arXiv preprint arXiv:1507.02646* (2015).

Aki Vehtari, Andrew Gelman, and Jonah Gabry. 2017. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and computing* 27, 5 (2017), 1413–1432.

Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. 2019. Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC. *arXiv preprint arXiv:1903.08008* (2019).

Sumio Watanabe. 2010. Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research* 11, Dec (2010), 3571–3594.