# Lumen: A software for the interactive visualization of probabilistic models together with data

Lucas, Philipp; Giesen, Joachim

Veröffentlichungsversion / Published Version
Zeitschriftenartikel / journal article

**Empfohlene Zitierung / Suggested Citation:**
Lucas, P., & Giesen, J. (2021). Lumen: A software for the interactive visualization of probabilistic models together with data. *The journal of open source software : a developer friendly journal for research software packages*, *63*(6), 1-4. https://doi.org/10.21105/joss.03395

The Journal of Open Source Software

# Lumen: A software for the interactive visualization of probabilistic models together with data

**Philipp Lucas**[*1] **and Joachim Giesen**[2]

**1** Institute of Data Science, German Aerospace Center **2** Friedrich-Schiller-University Jena

**DOI:** 10.21105/joss.03395

**Software**

- Review
- Repository
- Archive

**Editor:** Vissarion Fisikopoulos

**Reviewers:**

- @crhea93
- @szkafander

**Submitted:** 10 June 2021
**Published:** 17 July 2021



## Summary

Research in machine learning and applied statistics has led to the development of a plethora of different types of models. *Lumen* aims to make a particular yet broad class of models, namely, probabilistic models, more easily accessible to humans. *Lumen* does so by providing an interactive web application for the visual exploration, comparison, and validation of probabilistic models together with underlying data. As the main feature of *Lumen* a user can rapidly and incrementally build flexible and potentially complex interactive visualizations of both the probabilistic model and the data that the model was trained on.

Many classic machine learning methods learn models that predict the value of some target variable(s) given the value of some input variable(s). *Probabilistic* models go beyond this point estimation by predicting instead of a particular value a probability distribution over the target variable(s). This allows, for instance, to estimate the prediction's uncertainty, a highly relevant quantity. For a demonstrative example consider a model predicts that an image of a suspicious skin area does *not* show a malignant tumor. Here it would be extremely valuable to additionally know whether the model is sure to 99.99% or just 51%, that is, to know the uncertainty in the model's prediction.

*Lumen* is build on top of the *modelbase* back-end, which provides a SQL-like interface for querying models and its data (Lucas, 2020).

## Statement of need

A major challenge for both the application and development of machine learning/modelling methods is their accessibility to a human analyst, that is, the amount of hurdles that one must overcome to practically make use and benefit from them. *Lumen* aims to improve accessibility of probabilistic machine learning models with respect to multiple aspects as follows:

*Model Building:* Building a statistical/machine learning model is often an iterative, analyst-driven process. This is particularly true for the field of probabilistic programming, a modelling approach where the analyst explicitly declares the likelihood of the observed data as a probability density function. The analyst typically starts with an exploration of the data. Based on insights gained from data exploration and on the analyst's domain knowledge, the analyst creates an initial simple model involving only some data. Subsequently, this model is iteratively made more complex (Gabry et al., 2019; Gelman et al., 2013) until it meets the expert's goals. In particular, the model must be validated after each iteration. *Lumen* supports this model building process by (i) enabling visual-interactive data exploration, (ii) supporting model validation by means of a visual comparison of data queries to semantically equivalent model queries, and (iii) enabling a direct comparison of model iterates.

*corresponding author

*Debugging:* Even for a machine learning expert it may be hard to know whether a model has been trained on the data as expected. Possible reasons for artifacts in a model include an inappropriate application of the machine learning method, implementation bugs in the machine learning method, and issues in the training data. Direct visual inspection of the probabilistic model provides an approach to model debugging that enables the analyst to literally spot model artifacts that may cause degrading performance. Classical approaches to validation would rely on aggregating measures like information criterions or preditictive accuracy scores.

*Education:* By its intuitive visual representations of models, *Lumen* aims to promote understanding of the underlying modelling techniques. For instance, the effect of varying a parameter value for a modelling method on the probabilistic model can be observed visually rather than remaining an abstract description in a textbook. Similarily, the differences between models/model types can be visually illustrated by plotting them side by side. Also, probabilistic concepts such as conditioning or marginalization, which are often difficult to grasp, can be tried out interactively, providing immediate feedback.

## Software

*Lumen's* interface is inspired by the academic Polaris project and its commercial successor Tableau (Stolte et al., 2002). However, while Polaris/Tableau is for *data only*, *Lumen* provides a uniform visual language and interactions for both data and probabilistic models. Figure 1 shows an screenshot of *Lumen* to illustrate the user interface. The Schema panel (left) contains the random variables of the probabilistic model that the user has currently selected. Users can drag'n'drop variables onto the visual channels of the Specification panel (middle-left). This reconfigures the currently active visualization on the dashboard (middle to right), triggers execution of corresponding data and model queries, and finally updates and re-renders the visualization. To foster comparison of multiple models (for instance from different classes of models or from iterates of an incremental model building process) Lumen allows users to create as many visualizations of as many models as desired. All visualization support basic interactions like panning, zoom, or selections and are resizable as well as freely movable on the dashboard.

The Journal of Open Source Software


**Figure 1:** The Web-based interface of *Lumen* displaying a variety of visualizatons as created in the process of incrementally building a probabilistic model on the socio-economic ALLBUS data set (GESIS Leibniz-Institut für Sozialwissenschaften, 2017): (1) Marginal data density. (2) Marginal model density (pink) versus observed data density (grey). (3) Both plots show the same queries but from (a) to (b) the underlying model was improved to better capture the correlation of the `income` variable and the `sex` variable. Again, data are shown as histograms and model densities as line plots. (4) Connected dots show the model's point predictions of `income` given age and `sex`. Marks in the background as well as the marginal plots at the side represent observed data. (5) Similar to (4) but visualizing the model's predictions of `income` as well as of `happiness` given age and place of origin (`eastwest`). Again, the background marks show observed data.

While *Lumen* handles all user facing aspects (such as visualizations and interactions) most computational aspects (such as execution of model or data queries that are triggered by a user interaction) are delegated to a dedicated back-end. The back-end is implemented in the *modelbase* project (Lucas, 2020). This separation follows a classic client-server architecture where *Lumen* is the web-client and *modelbase* the web-service. For the standard usage scenario both client and server would be installed locally on the same machine. However, they can, of course, also be separated on different machines across a network.

*Lumen* is model-agnostic in the sense that it can be used with models of any class of probabilistic models as long as this model class implements the common, abstract API in the *modelbase* back end. The API essentially requires that a model class

- contains only quantitative and categorical random variables, i.e. Lumen has no native support for images, time series, or vector-valued random variables,
- supports marginalization of random variables, i.e. the operation to remove/integrate out any subset of random variables of the model,
- supports conditioning of random variables on values of its domain, i.e. the operation to fix the value of random variables to particular values, and
- supports density queries, i.e. the operation to ask for the value of the model's probability density function at any point of its domain.

In fact *Lumen* does not depend on any specific properties of a particular model class and we regard this genericity as one of *Lumens* major features. Among the model classes that we have used *Lumen* with are Sum-Product-Networks (Molina et al., 2019; Poon & Domingos, 2011), Condional-Gaussian Distributions (Nussbaum & Giesen, 2020; Olkin & Tate, 1961), Probabilistic Progams based on PyMC3 (Salvatier et al., 2016), and Kernel-Density-Estimators (Parzen, 1962; Virtanen et al., 2020).

## Acknowledgements

We thank Andreas Goral, Jonas Aaron Gütter, Laines Schmalwasser, Julien Klaus and Christian Lengert for their steady and patient interest in trying out Lumen, for their valuable feedback and our discussions, as well as for the features they contributed to Lumen. Philipp Lucas was partially supported by Stiftung der Deutschen Wirtschaft (sdw).

## References

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., & Gelman, A. (2019). Visualization in Bayesian workflow. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, *182*(2), 389–402. https://doi.org/10.1111/rssa.12378

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian Data Analysis* (3rd ed.). CRC Press. https://doi.org/10.1201/b16018

GESIS Leibniz-Institut für Sozialwissenschaften. (2017). *Allgemeine Bevölkerungsumfrage der Sozialwissenschaften ALLBUS 2016* (Version 2.1.0) [Data file]. GESIS Datenarchiv, Köln. https://doi.org/10.4232/1.12796

Lucas, P. (2020). Modelbase: A SQL-like interface for Python and the web to query probabilistic machine learning models and its data. In *GitHub repository*. GitHub. https://github.com/lumen-org/modelbase

Molina, A., Vergari, A., Stelzner, K., Peharz, R., Subramani, P., Mauro, N. D., Poupart, P., & Kersting, K. (2019). SPFlow: An easy and extensible library for deep probabilistic learning using sum-product networks. In *CoRR* (Vol. abs/1901.03704). http://arxiv.org/abs/1901.03704

Nussbaum, F., & Giesen, J. (2020). Pairwise sparse + low-rank models for variables of mixed type. *Journal of Multivariate Analysis*, *178*, 104601. https://doi.org/10.1016/j.jmva.2020.104601

Olkin, I., & Tate, R. F. (1961). Multivariate correlation models with mixed discrete and continuous variables. *Annals of Mathematical Statistics*, *32*(4), 448–465. https://doi.org/10.1214/aoms/1177705052

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, *33*(3), 1065–1076. https://doi.org/10.1214/aoms/1177704472

Poon, H., & Domingos, P. M. (2011). Sum-product networks: A new deep architecture. *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*, 337–346. http://arxiv.org/abs/1202.3732

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, *2*, e55. https://doi.org/10.7717/peerj-cs.55

Stolte, C., Tang, D., & Hanrahan, P. (2002). Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, *8*(1), 52–65. https://doi.org/10.1109/2945.981851

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2